République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
du diplôme d'ingénieur d'état en informatique.
Option : Système d'information

Sujet :

Réalisation d'un outil d'aide à la conception et l'implémentation des zones démilitarisées (DMZ)

Présenté par : BOUDAOUD Abdelmalek
FAREH Abdelkader

Promoteur : Mr. ZAHAR Cherif
Encadreur : Mr. BENMOUSSA Yahia

Organisme d'accueil : Naftal.

Soutenue le: date soutenance, devant le jury composé de :

Nom président du jury, grade, organisme	**Président**
Nom examinateur 1, grade, organisme	**Examinateur**
Nom examinateur 2, grade, organisme	**Examinateur**

-2006/2007-

Remerciement

Avant toute chose, le véritable remerciement revient au grand Dieu tout puissant qui nous a donné la force de réaliser ce travail.

Nous remerciant notre promoteur Mr. ZAHAR Cherif pour sa disponibilité et ses conseils, et notre encadreur Mr. BENMOUSSA Yahia pour avoir bien voulu nous proposer ce sujet et pour sa patience et son soutient continue durant notre stage.

Nous remercions également tous les professeurs de USDB qui nous ont accompagnés durant nos études supérieures, ainsi qu'à tous nos enseignants et enseignantes d'étude primaire, moyenne et lycéenne.

Nous remercions tous ceux qui ont contribué à la réalisation de ce travail de prés ou de loin, matériellement ou moralement, et partager nos peines pour voir ce travail naître. De même exprimons nous notre reconnaissance à l'ensemble des étudiants de l'institut d'informatique de la promotion 2006/2007.

Dédicaces

Je dédie ce modeste travail

A qui sont très chère à mon cœur,

Ma mère et mon père

A mes chère frères Mohamed, Abdelrrazak, et Hamza.

A toutes mes chers sœurs.

Et à toute ma grande famille sans exception

A mon binôme et sa famille.

Et à tous mes amis

Abdelkader

Dédicaces

Je dédie ce modeste travaille :

A qui est très chère à mon coeur, ma merveilleuse mère pour son amour.

A qui a sacrifié sa vie pour nous, mon cher père.

A mes chers frères Noureddine, Hedi, Mehdi et Azzeddine.

A ma petite fleure Wafaa.

A ma tante Lwiza.

Et à toute ma grande famille sans exception.

A mon binôme et sa famille.

A tout les gens qui m'ont aimé, à tous mes amis et collègues sans restriction.

A Mr. Benmoussa Yahia et Mr. Zahar Cherif.

Abdemalek

RESUME

L'outil développé dans ce projet de fin d'étude permet la mise en place d'une politique de sécurité du réseau de l'entreprise basée sur le concept **DMZ (DeMilitarized Zone).** Ce dernier peut être vu comme une boite noire acceptant en entrée un programme source saisi par l'utilisateur et qui renvoie en sortie la liste de commandes **IPtables** qui lui sont équivalentes.

Le programme d'entrée doit respecter une syntaxe bien définie.

La liste des commandes **IPtables** générées automatiquement par l'outil, sert à la configuration du **Firewall Netfilter.** Cette configuration correspond à l'architecture de sécurité en DMZ que nous avons proposée.

Mots clés :

Sécurité de réseaux, pare-feu, IPtables, Netfilter, DMZ (zone démilitarisée),

ABSTRACT

The tool developed in this project, putting into service a network security politic for a commerce firm, based of the **DMZ**, we can see this tool as a Black box which has a source program in the input, and **IPtables** controls in the output abide by the input source.

The input program mast follow define syntax.

The list of IPtables controls, generate automatically by this tool, use for the configuration of **Firewall Netfilter.**

Key words:

Network security, firewall, IPtables, Netfilter, DMZ (demilitarized zone).

ملخـــــــــص

الـــوســـيلة المطورة في هذا الـــمشروع (مشروع نـــهاية الـــدراسة), تمكن من توفـــير حماية للشبكة الداخلية للـــمؤسسة عن طريق تصميم لما يسمى بالحيز منزوع السلاح (DMZ), و يمكن رؤيـــة هذه الوسيـــلة كـــعلبة سوداء, تـــقبل في الدخول برنامج مصدر منجـــز من طرف المستـــعمل لهـــذه الوسيلة, و تنتج في الخروج أوامر IPtables الـــتي توافق برنامج الدخول.

- يجب أن يحترم بـــرنامج الدخول نحو معرف مسبـــقا.
- قائـمة أوامر IPtables الناتجة آليا عن هذه الـوسـيلة, تـمكن من تهـيئة أو برمجة الـحائط الـناري (Firewall Netfilter).

مـفـاتـــيح:
حماية الشبكة, حائط النار, IPtables, Netfilter, حيز منزوع السلاح.

Sommaire

Introduction générale 1
Présentation de la société 2

Partie I : Etat de l'art

Chapitre 1 : Présentation de TCP-IP

1.1. Introduction 7
1.2. Archétecture du protocole TCP-IP 7
1.3. Hierarchie et implementation de TCP/IP 9
1.4. Les adresses IP 10
1.5. Les masques de sous réseau 13
1.6. La notion de port d'écoute 14
1.7. Le domaine de noms 14
1.8. Conclusion 14

Chapitre 2 : Introduction au Pare-feu (Firewall)

2.1. Introduction 16
2.2. La sécurité des réseaux 16
2.2.1. Qu'est-ce que la sécurité d'un réseau ? 16
2.2.2 Les types d'insécurité 16
2.2.3. Le but des agresseurs 16
2.2.4. Procédé des agresseurs 17
2.2.5. Comment se protéger ? 17
2.3. Pare-feu (Firewall) 17

2.3.1. Qu'est-ce qu'un Pare-feu? 18
2.3.2 Fonctionnement d'un système Pare-feu 19
2.3.3. Fonctionnement du Pare-feu sous Linux 23
2.4. Conclusion 25

Chapitre 3 : Etude du Firewall Netfilter

3.1. Introduction 27
3.2. Fonctionnement général de Netfilter 27
3.3. Description des commandes IPtables 31
3.4. Conclusion 33

Chapitre 4 : Etude des DMZs (Demilitarized Zone)

4.1. Introduction 35
4.2. Définitions 36
4.3. Firewall et DMZ ? 37
4.4. Description du flux d'échange entre les différentes entités.. 39
4.5. Conclusion 40

Partie II : Conception

Chapitre 1 : Proposition d'un formalisme pour la modélisation des DMZs

1.1. Problématique 43
1.2. Solution proposée 43
1.3. L'environnement de développement 46
1.4. l'analyse lexical avec Lex 46
1.5. L'analyse grammaticale avec Yacc 47

Chapitre 2 : Conception du préprocesseur à réaliser

2.1. Introduction **49**

2.2. Description du langage du préprocesseur à réaliser **49**

2.6. Conclusion **52**

Partie III : Implémentation

Chapitre 1 : Programmation du préprocesseur

1.1. Introduction **55**

1.2. Description des fichiers Lex et Yacc **55**

1.2.1. Les différentes parties du fichier Lex **55**

1.2.2. Les différentes parties du fichier Yacc **56**

1.4. Schéma de compilation **67**

1.5. Conclusion **68**

Chapitre 2 : Tests du préprocesseur

2.1. Introduction **70**

2.3. Tests et résultats **70**

2.8. Conclusion **77**

Annexe :

Annexe A : Utilisation de Lex et Yacc **81**

Annexe B : Manuel d'utilisation **88**

Liste des figures :

Figure 1 : L'organigramme de NAFTAL..... 4

Partie I : Chapitre1

Figure I.1.1 : Architecture de TCP/IP..... 8

Figure I.1.2 : Encapsulation des données..... 8

Figure I.1.3 : Plan d'adressage IP..... 12

Figure I.1.4 : Un réseau de classe B subdivisé en 64 sous réseaux..... 13

Partie I : chapitre2 :

Figure I.2.1 : Représentation d'un système Firewall..... 18

Figure I.2.2 : Les deux interfaces d'un Pare-feu..... 24

Figure I.2.3 : La structure des différentes chaînes du Pare-feu Netfilter..... 25

Partie I : chapitre3 :

Figure I.3.1 : Topologie de la machine Linux..... 27

Figure I.3.2 : L'acheminement des paquets IP à travers les différentes chaînes de Netfilter..... 28

Figure I.3.3 : La syntaxe générale des commandes IPtables..... 31

Partie I : chapitre4 :

Figure I.4.1 : Architecture d'un réseau d'entreprise disposant d'une DMZ isolée par un Firewall..... 35

Figure I.4.2 : Une DMZ protégée par le Firewall..... 37

Figure I.4.3 : Une DMZ protégée par un routeur sur lequel est installé un logiciel de contrôle..... 38

Partie II : chapitre1 :

Figure II.1.1 : Le schéma de DMZ considéré et les différents flux échangés..... 44

Partie III: chapitre1:

Figure III.1.1: Le schéma de compilation du source Lex et Yacc..... 67

Liste des tables :

Table I.1.1 : La hiérarchie d'implémentation de TCP/IP 9

Table I.2.1 : Exemples de règles de Pare-feu..... 20

Table I.3.1 : Les actions possibles des règles IPtables..... 32

Table IV : La manière d'écriture des expressions régulières..... 83

Introduction générale :

Avec le poids économique de plus en plus important de l'informatique dans l'industrie, en plus du fait de la dépendance grandissante envers les réseaux de communication, et étant donnée le grand nombre de réseaux interconnectés, la sécurité des réseaux informatiques est aujourd'hui un problème crucial. Ce problème est aggravé avec l'existence des attaques externes, surtout après l'apparition de l'Internet.

Mais les efforts de sécurisation ne peuvent avoir d'effet que si ces investissements sont correctement ciblés et étudiés, avec la mise en place des moyens de protection apportant un niveau de sécurité adapté aux enjeux spécifiques de l'entreprise.

La définition d'une politique de sécurité informatique vise ainsi à identifier les besoins de l'organisation en termes de sécurité et à élaborer des stratégies afin de protéger les biens de l'entreprise.

L'entreprise nationale **NAFTAL** est consciente des conséquences de l'absence de sécurité vue qu'elle dispose d'un réseau vaste, ce qui nécessite d'avoir une sécurité d'un degré élevé pour pouvoir garantir la disponibilité de ces données.

Vu que le système d'exploitation utilisé dans les différents services réseau de l'entreprise est **Linux,** l'implémentation de la politique de sécurité est faite par la configuration du Firewall **Netfilter** pour la mise en œuvre de la **DMZ** via la commande **IPtables,** le problème rencontré est que l'administrateur du réseau trouve des difficultés à administrer **Netfilter** dès que le nombre de règles de filtrage augmente.

Notre objectif est la réalisation d'un outil d'aide à la conception et l'implémentation des zones démilitarisées. Ce dernier est un **préprocesseur** permettant de remplacer l'usage compliqué des commandes **IPtables** par un pseudo langage simple. Le source écrit dans ce dernier langage sera traduit par le **préprocesseur** en une liste de commandes **IPtables** équivalentes.

Présentation de la société:

NAFTAL, Société par actions filiale à 100% de Sonatrach, a pour mission principale la distribution et la commercialisation des produits pétroliers sur le marché national.

NAFTAL a mis en place une nouvelle stratégie de développement dont l'objectif est de conserver sa position de leader de la distribution sur le plan national et de se déployer sur le plan international.

Dans ce contexte, 04 NAFTAL a mis en place une nouvelle organisation, composée de six divisions opérationnelles ayant une totale indépendance de gestion, en l'occurrence la division Carburant, la division commerciale, la division lubrifiants et pneumatiques, la division GPL, la division Aviation/Marine et enfin la division bitumes.

La société contribue à hauteur de 51% de l'énergie finale algérienne en fournissant plus de 9 millions de tonnes tous produits confondus en l'occurrence, les carburants, les GPL, les bitumes, les lubrifiants, les produits aviation/marine et les produits spéciaux. NAFTAL a, en outre, lancé un large programme de formation à l'endroit de ses employés et ses cadres, une modernisation de ses installations et de ses infrastructures ainsi qu'une mise a niveau de tous ses systèmes de gestion.

NAFTAL a inscrit dans sa stratégie de développement une option sérieuse pour l'international et elle est déjà présente sur le marché international puisqu'elle réalise du remplissage à partir de Tébessa vers la Tunisie; elle est également présente en Mauritanie. NAFTAL est en discussion avec des groupes privés Marocain, Egyptien, Libyen et Yéménite pour des alliances stratégiques à travers la création de sociétés mixtes, des joint-ventures ou des échanges d'actifs.

Elle est aussi en discussion avec plusieurs sociétés de pays africains, européens et latino américains.

Dans ce contexte, NAFTAL est en pourparlers avec des majors de la distribution des produits pétroliers à l'instar de BP, Shell et Agip. Pour la période 2003-2007, NAFTAL s'est engagée sur un programme de développement qui nécessite une enveloppe financière de 35 millions USD. 70% des fonds mobilisés seront orientés vers l'activité de distribution stratégique: 3 pipelines, stockage carburant, 1 centre enfûteur (25M $), infrastructures GPL; les 30% restants seront destinés à l'activité de la petite et moyenne distribution: Stations services,

flotte de transport, infrastructures de maintenance ainsi que l'introduction de systèmes de gestion modernes.

L'ensemble des installations seront améliorées pour une meilleure mise en conformité avec la nouvelle législation relative à la protection de l'environnement pour laquelle NAFTAL accorde la plus grande importance à travers, notamment le développement de l'essence sans plomb, des GPL et la récupération des huiles usagées.

1. Historique:

Issue de SONATRACH, l'entreprise ERDP a été créée par le décret N° 80/101 du 06 avril **1981**.

Entrée en activité le 1er janvier **1982**, elle est chargée de l'industrie du raffinage et de la distribution des produits pétroliers sous le sigle NAFTAL.

En **1987**, l'activité raffinage est séparée de l'activité distribution.

La raison sociale de la société change suite à cette séparation des activités et NAFTAL est désormais chargée de la commercialisation et de la distribution des produits pétroliers et dérivés.

A partir de 1998, elle change de statut et devient Société par actions filiale à 100% de SONATRACH.

NAFTAL a pour mission principale, la distribution et la commercialisation des produits pétroliers sur le marché national.

Elle intervient dans les domaines :

- *de l'enfûtage GPL*
- *de la formulation de bitumes*
- *de la distribution, stockage et commercialisation des carburants, GPL, lubrifiants, bitumes, pneumatiques, GPL/carburant, produits spéciaux.*
- *du transport des produits pétroliers*

2. Organisation:

Suite à son intégration dans le groupe Sonatrach dont elle est filiale à 100%, NAFTAL s'est réorganisée autour de quatre (04) Branches.

Les divisions ont pour mission de définir avec la direction générale, la stratégie de distribution et de commercialisation des produits pétroliers en veillant à rassembler toutes les conditions de son application dans les centres opérationnels de la société.

Schéma actuel d'organisation de la société
NAFTAL

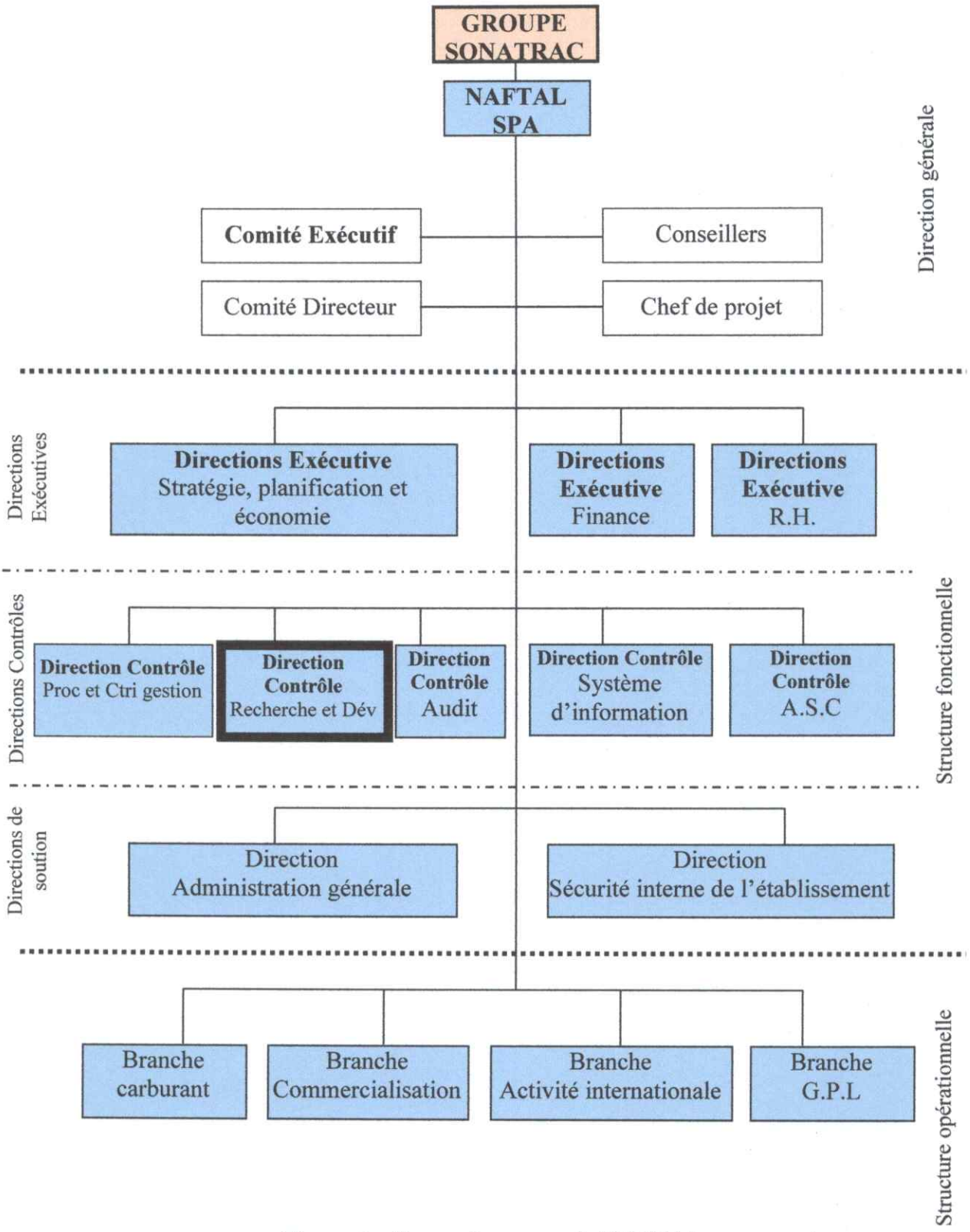


Figure 1 : l'organigramme de NAFTAL

La direction où on a suivi notre projet de fin d'étude.

Partie I

Etat de l'art

Chapitre 1

Présentation de TCP-IP

Dans ce chapitre :

- Introduction
- Architecture du protocole TCP/IP
- Hiérarchie et implémentation de TCP/IP
- Les Adresses IP
- Les masques de sous réseau
- La notion de port d'écoute
- Le domaine de noms
- Conclusion

1.1. Introduction:

Le protocole **TCP/IP** (transport control protocol / Internet Protocole), est toute une suite de protocoles de communication dans les réseaux, recouvrant deux protocoles majeurs, **TCP** et **IP**, d'où vient son nom. Le protocole TCP/IP représente une façon avec laquelle les ordinateurs d'un réseau peuvent communiquer entre eux. La suite des protocoles TCP/IP était crée à l'origine d'un besoin militaire pour répondre à certains critères tels que le fractionnement des messages en paquet, l'utilisation d'un système d'adressage des ordinateurs, l'acheminement des données sur le réseau, etc...

1.2. Architecture du protocole TCP/IP : [R1.1]

Le système de protocoles **TCP/IP** à été décomposée en plusieurs modules effectuant chacun un rôle précis. De plus ces modules effectuent des taches les unes après les autres dans un ordre précis pour chaque machine et indépendamment du système d'exploitation, on à donc un système stratifié, c'est la raison pour laquelle on parle de modèle en couches.

Les deux modèles décrivant les couches de protocoles sont les suivants :

- Le modèle ***DOD*** (Departement Of Defence) qui comprend 4 couches.
- Le modèle ***OSI*** (Open System Interconnexion) qui comprend 7 couches.

Dans se qui suit nous allons décrire seulement le modèle ***DOD*** :

Le schémas suivant montre la description des quatre couches composant le modèle ***DOD.***

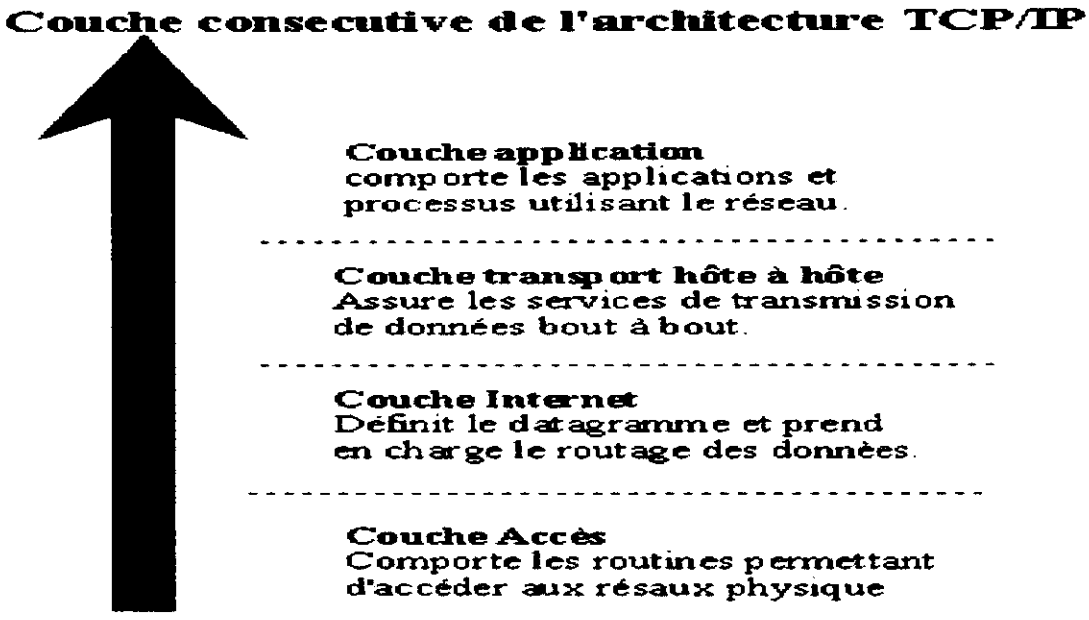


Figure I.1.1 : Architecture de TCP/IP.

Couche Accès (physique) :

- La couche physique correspond en gros aux couches 1 et 2 de l'OSI. Elle représente la connexion physique avec les câbles, les cartes réseaux, et les protocoles d'accès aux réseaux.

Couche Internet (IP) :

- La couche **IP** correspond au niveau 3 de l'OSI, elle fournit l'adresse logique pour l'interface physique. On envoie un datagramme et on « espère » qu'il arrive. Les contrôles d'erreur, contrôles de flux.., sont donc à la charge des couches supérieures.

Couche Transport Hôte à Hôte (TCP) :

- La couche **TCP** correspond à la couche 4 de l'OSI (transport de bout en bout), assure les connexions entre les différents hôtes sur le réseau.

Couche Application :

- Enfin, la couche application (équivalente aux couches 5, 6 et 7 de l'OSI), comporte un certain nombre d'applications standardisées qui s'appuient elles-mêmes sur **TCP** ou **UDP**.

Chaque couche de la pile ajoute des informations de contrôle de manière à garantir une transmission de données correcte.

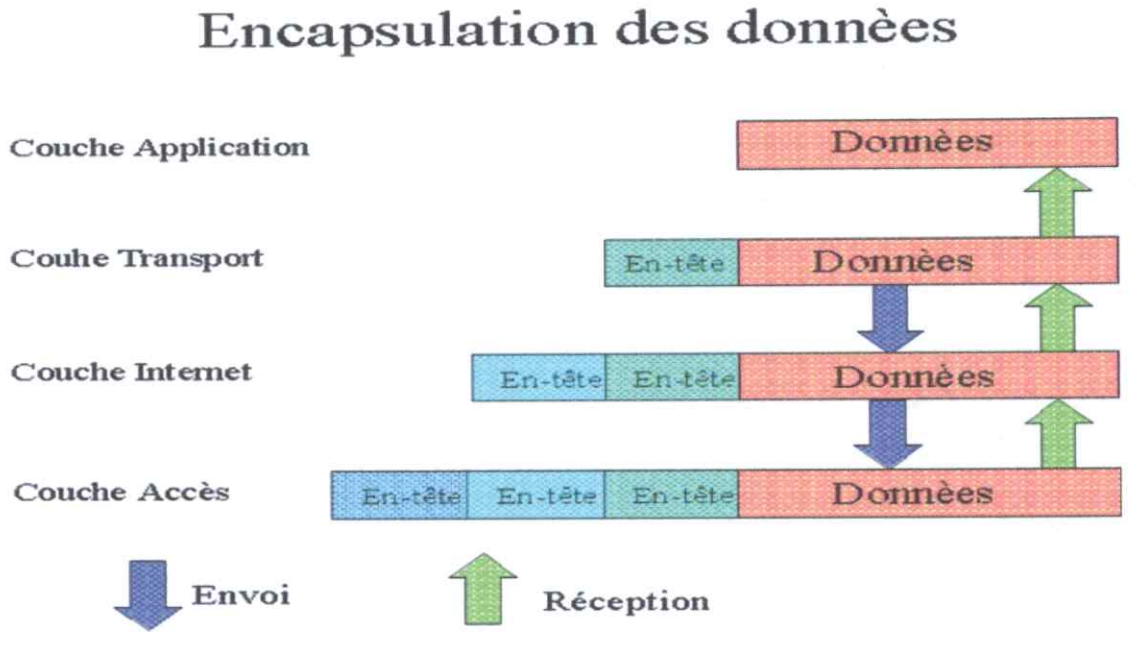


Figure I.1.2 : Encapsulation des données.

Lors d'une émission d'un message par un hôte, il traverse chacune des couches du modèle ***DOD,*** chaque couche ajoute une entête au paquet de données reçu.

A la réception, chaque couche lit le paquet de donnée reçu et élimine l'entête approprié, et le message prend son état original.

1.3. Hiérarchie et implémentation de TCP/IP :

La famille des protocoles **TCP/IP** comprend un ensemble performant de services qui peuvent utiliser une variété de technologies réseaux (réseaux étendus ou locaux, les ondes radio, les satellites,…). Les modules du **TCP/IP** et les relations entre ces modules forment la hiérarchie de l'implémentation **TCP/IP**.

Table I.1.1: La hiérarchie d'implémentation de TCP/IP :

Couche 1 (application)	**HTTP**	**FTP**	**SMTP**	**DNS**	**IMAP**	**POP3**	**NNTP**	**…**
Couche 2 (transport)	**TCP**				**UDP**			
Couche 3 (Internet)	**IP**		**ICMP**	**RIP**		**ARP**	**RARP,...**	
Couche 4 (accès réseau)	**Ethernet, Token Ring, FDDI, ATM, PPP,...**							

Cette table montre les quatre modules suivants :

- **Le module 1 :** représenté par la couche 4, il se charge des protocoles d'accès réseau.
- **Le module 2 :** représenté par la couche 3, il se charge des protocoles **IP** (pour l'acheminement des paquets depuis l'expéditeur vers le distinataire, **ICMP** (Internet Control Message Protocol, pour la gestion des messages et des erreurs), **RIP** (Routing Information Protocol, pour la détermination des routes des paquets), **ARP** (Address Résolution Protocol, pour la translation des adresses **IP** en adresse physique) et **RARP** (Reverse **ARP**).
- **Le module 3 :** représenté par la couche 2, il se charge des protocoles **TCP** (qui s'appuie sur **IP** pour le transfert des données entre l'expéditeur et le distinataire en établissant des connexions), et **UDP** (User Datagram Protocol, pour le transfert de données entre l'expéditeur et le distinataire sans connexion).
- **Le module 4 :** représenté par la couche 1, et se charge des services réseaux (**DNS, HTTP, FTP, ...**).

1.4. Les Adresses IP : [R1.2]

TCP/IP utilise un système d'adressage permettent un routage simple des datagrammes à travers le réseau Internet. Chaque machine d'une interconnexion **TCP/IP** dispose d'une adresse **IP** unique représentée par 4 entiers séparés d'un point décimal. (Valeur comprise entre 0 et 255 pour chaque octets, ex: 165.3.5.2).

Le protocole Internet transmet des données entre hôtes sous la forme de datagrammes. Chaque datagramme est transmis à l'adresse contenue dans le cinquième mot de l'en-tête (Adresse destination) sur 32 bits contenant les informations suffisantes pour identifier un réseau et une machine hôte déterminée de celui-ci.

Une adresse IP comporte deux parties: la "partie réseau" et la "partie hôte". Le format de ces parties différent d'une adresse IP à l'autre. Le nombre de bits d'adresse utilisé pour identifier le réseau et le nombre utilisé pour reconnaître l'hôte varie en fonction de " la classe d'adresse".

Les classes d'adresses

Les trois principales classes d'adresses sont la classe A, la classe B et la classe C. L'examen des premiers bits de l'adresse permet au logiciel IP de déterminer la classe de l'adresse.

La classe A

Si le premier bit d'une adresse est positionné sur **0**, il s'agit alors d'une adresse réseau de classe A. Le premier bit d'une adresse de classe A permet d'identifier la classe de l'adresse. Les 7 bits suivants permettent de déterminer le réseau et les 24 derniers bits de reconnaître l'hôte. Il existe moins de 128 numéros de réseau de classe A, chaque réseau de classe A peut être constitué de millions d'hôtes.

La classe B

Si les 2 premiers bits de l'adresse sont **10**, il s'agit alors d'une adresse de réseau de classe B. Les 2 premiers bits permettent de déterminer la classe; les 14 bits suivants de déterminer le réseau. Les 16 derniers bits permettent de reconnaître l'hôte. Il existe des milliers de numéros de réseau de classe B et chaque réseau de classe B peut-être constitué de milliers d'hôtes.

La classe C

Si les 3 premiers bits de l'adresse sont **110**, il s'agit alors d'une adresse de réseau de classe C. Dans une adresse de classe C, les 3 premiers bits sont des identificateurs de classe; les 21 bits suivants correspondent à l'adresse du réseau et les 8 derniers bits permettent d'identifier l'hôte. Il existe des millions de numéros de réseau de classe C. Chaque réseau de classe C comporte moins de 254 hôtes.

La classe D

Si les 3 premiers bits de l'adresse sont **111**, il s'agit d'une adresse spéciale réservée. Ces adresses sont parfois appelées adresse de classe D, mais elles ne correspondent pas à des réseaux spécifiques. Les numéros attribués dans cette classe correspondent à des adresses multidestinataires. Les adresses multidestinataire sont utilisées pour adresser des groupes d'ordinateurs simultanément, ils permettent aussi d'identifier un groupe d'ordinateurs

partageant un protocole commun, par opposition à un groupe d'ordinateurs partageant un réseau commun, la classe D sert au mécanisme de diffusion de groupe IGMP.

Une machine utilisant TCP/IP possède une ou plusieurs adresses IP qui sont indépendantes des adresses matérielles.

Les programmes d'application utilisent toujours les adresses IP pour indiquer une destination (indépendance vis à vis du matériel).

Plan d'adressage du domaine IP

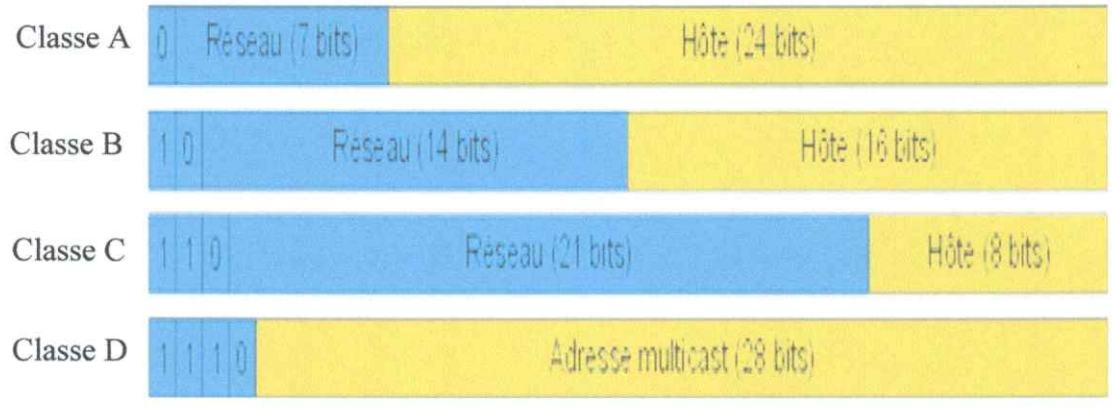


Figure I.1.3: Plan d'adressage IP.

- Les machines et les passerelles utilisent les adresses physiques pour transmettre des données sur un réseau donné.

- Dans le cas où une machine désire communiquer avec une autre machine ne situant pas sur le même réseau physique. L'adresse physique utilisée est l'adresse de la passerelle qui relie les 2 réseaux.

Convention

- ✓ L'adresse du réseau a une valeur d'indentification de machine où tous les bits sont nuls (0). (ex: adresse réseau classe B 163.5.0.0)
- ✓ Une adresse contenant les bits de réseau à 0 signifie ce réseau. (ex: 0.0.0.23, machine 23 sur ce réseau)

- ✓ Une adresse de diffusion « restreinte » ou « dirigée » contient un identificateur de machine où tous bits sont à 1. (ex: adresse réseau classe B 163.5.255.255 BroadCast)
- ✓ Une adresse de diffusion « locale » ou « plein 1 » (« all 1s ») comporte les 32bits IP à 1. (Utiliser lors du démarrage)
- ✓ L'adresse de Classe A de « boucle locale » (loopback) vaut 127.0.0.1. Elle est destinée à permettre les communications inter-processus sur ma machine locale.

1.5. Les masques de sous réseau :

Pour laisser un maximum de souplesse du mode de décomposition de l'adresse IP en sous réseaux, la norme TCP/IP du sous adressage permet de choisir l'interprétation des sous réseaux indépendamment pour chaque réseau. Une fois qu'une décomposition en sous-réseaux a était choisie, toutes les machines du réseau doivent s'y conformer

La création de sous-réseaux divise une adresse réseau en plusieurs adresses de sous-réseaux uniques, de sorte qu'une adresse spécifique puisse être attribué à chaque réseau physique.

L'application d'un masque de bit, appelé masque de sous-réseau, à l'adresse IP permet de définir un sous-réseau. Si un bit déterminé est activé dans le masque (mis à 1), ce bit est considéré comme un bit de réseau, sinon il s'agit d'un bit appartenant à la partie de la machine.

Les masques de sous réseaux sont écrits soit en notation décimale pointée, soit en notation préfixée, c'est-à-dire sous la forme d'une barre oblique suivie du nombre de bites octroyés aux portions réseau et sous réseaux combinées. Le masque de la figure I.1.4 peut donc s'écrire des deux façons suivantes : 255.255.252.0 ou /22.

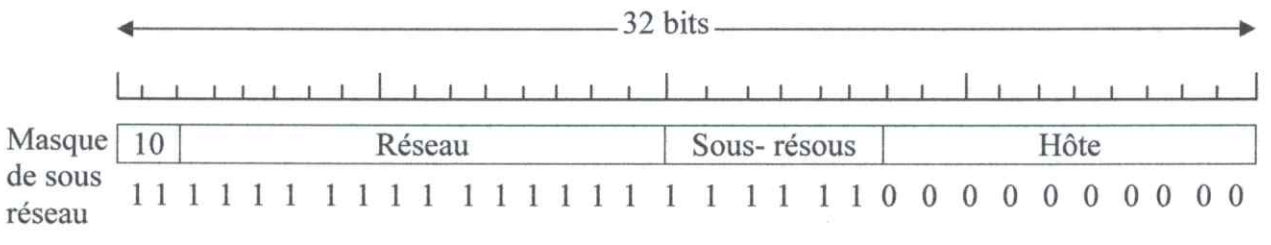


Figure I.1.4 : un réseau de classe B subdivisé en 64 sous réseaux.

Pour un réseau de classe B, le masque de sous réseau est: 255.255.0.0.

1.6. La notion de port d'écoute :

Lorsque les applications utilisent les services réseaux tels que **HTTP, FTP,...** qui doivent communiquer sur Internet ou sur un réseau **TCP/IP,** elles utilisent un paramètre supplémentaire en plus de l'adresse IP de l'expéditeur et du distinataire : le numéro de port du service (chaque service à son propre numéro de port). Chaque paquet transmis sur le réseau contient un champ indiquant le numéro de port du service demandé par l'application.

Certains numéros de port sont usuels (well-known, compris entre 0 et 1023 et sont réservées a l'IANA, Internet Address Network Autority), et d'autres sont affectés dynamiquement (les numéros de ports non enregistrés). Par exemple le port **TCP** 80 est un numéro de port usuel et qui est utilisé par le protocole **HTTP.**

1.7. Le domaine de noms :

TCP/IP comme nous avons vu, utilise l'adresse **IP** qui est un numéro sur 32 bits pour acheminer un datagramme à une distination donnée. Il est utile d'oublier ces numéros d'adresse et utiliser des noms communs dans les différentes applications basées sur **TCP/IP** parce que les noms sont plus faciles à se souvenir.

Il y'a plusieurs méthodes utilisées pour supporter l'utilisation du domaine de noms en **TCP/IP.** La plus ancienne est d'employer un fichier texte sur la machine émettrice qui avait des noms et des adresses correspondantes (/etc/hosts sur une machine linux), mais la plus efficace est d'utiliser le service de noms **DNS** (Domain Name Service).

1.8. Conclusion :

Une intervention sur la gestion, l'administration ou la configuration des réseaux sous Linux ou n'importe quel produit UNIX, se répercute inévitablement sur les modules **TCP/IP.**

Les modules **TCP/IP** sont construits hiérarchiquement à partir d'une suite de protocoles disposés en couches, ces modules sont connus souvent par les couches de protocoles.

Dans les chapitres suivants nous allons décrire le système à configurer par notre outil.

Chapitre 2

Introduction au Pare-feu (Firewall)

Dans ce chapitre :

- Introduction
- La sécurité des réseaux
- Pare-feu (Firewall)
- Conclusion

2.1. Introduction :

La sécurité des systèmes informatiques comprend trois aspects majeurs qui sont : la confidentialité, l'intégrité, et la disponibilité du service. Pour assurer ces trois aspects il existe plusieurs approches, l'approche la plus utilisée est la mise en place d'un mécanisme de filtrage placée au niveau des points d'accès au réseau. Ce mécanisme de filtrage est appelé **Firewall.**

2.2. La sécurité des réseaux : [R2.1]

2.2.1. Qu'est-ce que la sécurité d'un réseau ?

La sécurité d'un réseau est un niveau de garantie que l'ensemble des machines du réseau fonctionnent de façon optimale et que les utilisateurs des machines possèdent uniquement les droits qui leurs ont été octroyés.

Il peut s'agir :

- d'empêcher des personnes non autorisées d'agir sur le système de façon malveillante ;
- d'empêcher les utilisateurs d'effectuer des opérations involontaires capables de nuire au système ;
- de sécuriser les données en prévoyant les pannes ;
- de garantir la non-interruption d'un service.

2.2.2. Les types d'insécurité

On distingue généralement deux types d'insécurité :

- **l'état actif d'insécurité**, c'est-à-dire la non-connaissance par l'utilisateur des fonctionnalités du système, dont certaines pouvant lui être nuisibles (par exemple la non-désactivation de services réseaux non nécessaires à l'utilisateur).
- **l'état passif d'insécurité**, c'est-à-dire lorsque l'administrateur (ou l'utilisateur) d'un système ne connaît pas les dispositifs de sécurité dont il dispose.

2.2.3. Le but des agresseurs

Les motivations des agresseurs que l'on appelle communément "pirates" peuvent être multiples :

- l'attirance de l'interdit ;
- le désir d'argent (violer un système bancaire par exemple) ;
- le besoin de renommée (impressionner des amis) ;
- l'envie de nuire (détruire des données, empêcher un système de fonctionner).

2.2.4. Procédé des agresseurs

Le but des agresseurs est souvent de prendre le contrôle d'une machine afin de pouvoir réaliser les actions qu'ils désirent. Pour cela il existe différents types de moyens :

- l'obtention d'informations utiles pour effectuer des attaques ;
- utiliser les failles d'un système ;
- l'utilisation de la force pour casser un système.

2.2.5. Comment se protéger ?

- se tenir au courant ;
- connaître le système d'exploitation ;
- **réduire l'accès au réseau (Firewall) ;**
- réduire le nombre de points d'entrée (ports) ;
- définir une politique de sécurité interne (mots de passe, lancement d'exécutables) ;
- déployer des utilitaires de sécurité (journalisation).

2.3. Pare-feu (Firewall):

Chaque ordinateur connecté à Internet (et d'une manière plus générale à n'importe quel réseau informatique) est susceptible d'être victime d'une attaque d'un pirate informatique. La méthodologie généralement employée par le pirate informatique consiste à scruter le réseau (en envoyant des paquets de données de manière aléatoire) à la recherche d'une machine connectée, puis à chercher une faille de sécurité afin de l'exploiter et d'accéder aux données s'y trouvant. **[R2.2]**

Cette menace est d'autant plus grande que la machine est connectée en permanence à Internet pour plusieurs raisons :

- La machine cible est susceptible d'être connectée sans pour autant être surveillée ;
- La machine cible est généralement connectée avec une plus large bande passante ;

- La machine cible ne change pas (ou peu) d'adresse IP.

Ainsi, il est nécessaire, autant pour les réseaux d'entreprises que pour les internautes possédant une connexion de type câble ou ADSL, de se protéger des intrusions réseaux en installant un dispositif de protection.

2.3.1. Qu'est-ce qu'un Pare-feu?

Un **Pare-feu** (appelé aussi ***coupe-feu***, ***garde-barrière*** ou **Firewall** en anglais), est un système permettant de protéger un ordinateur ou un réseau d'ordinateurs des intrusions provenant d'un réseau tiers (notamment Internet). Le Pare-feu est un système permettant de filtrer les paquets de données échangées avec le réseau, il s'agit ainsi d'une passerelle filtrante comportant au minimum les interfaces réseau suivantes :

- une interface pour le réseau à protéger (réseau interne) ;
- une interface pour le réseau externe.

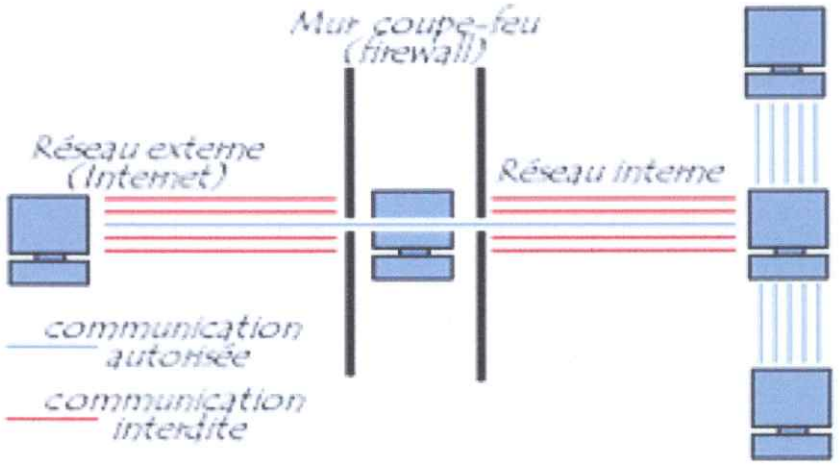


Figure I.2.1 : Représentation d'un système Firewall.

Le système Firewall est un système logiciel, reposant parfois sur un matériel réseau dédié, constituant un intermédiaire entre le réseau local (ou la machine locale) et un ou plusieurs réseaux externes. Il est possible de mettre un système Pare-feu sur n'importe quelle machine et avec n'importe quel système pourvu que : **[R2.2]**

- La machine soit suffisamment puissante pour traiter le trafic ;
- Le système soit sécurisé ;
- Aucun autre service que le service de filtrage de paquets ne fonctionne sur la machine.

2.3.2. Fonctionnement d'un système Pare-feu :

Un système pare-feu contient un ensemble de règles prédéfinies permettant :

- D'autoriser la connexion (***allow***) ;
- De bloquer la connexion (***deny***) ;
- De rejeter la demande de connexion sans avertir l'émetteur (***drop***).

L'ensemble de ces règles permet de mettre en oeuvre une méthode de filtrage dépendant de la **politique de sécurité** adoptée par l'entité. On distingue habituellement deux types de politiques de sécurité permettant :

- soit d'autoriser uniquement les communications ayant été explicitement autorisées :

  "Tout ce qui n'est pas explicitement autorisé est interdit".

- soit d'empêcher les échanges qui ont été explicitement interdits.

La première méthode est sans nul doute la plus sûre, mais elle impose toutefois une définition précise et contraignante des besoins en communication.

Notion de filtrage de paquets IP :

Une adresse IP permet l'identification de manière unique d'un ordinateur connecté a Internet, celle-ci est constituée d'une partie correspondant au numéro du réseau et d'une étant le numéro de la machine dans ce réseau.

Seul l'en-tête IP d'une trame peut laisser des traces lors de son passage, les rubriques utiles pour le filtrage de paquets sont :

- Types de paquets (TCP, UDP…) ;
- Adresse IP d'origine ;
- Adresse IP de destination ;
- Le port de destination, et d'origine (TCP, UDP…).

Les filtres de paquets travaillent en triant les paquets d'après leurs adresses ou leurs ports d'origine ou de destination. En général, aucun contexte n'est réservé, les décisions sont prises seulement d'après le contenu du paquet en cours de traitement. L'administrateur fait une liste

de services et machines irrecevables. Il est facile de permettre ou de refuser l'accès au niveau du réseau ou des machines avec un filtre de paquets.

Les fonctions de filtrages :

Il existe trois fonctions de filtrage, qui sont [CHA 2.1] :

- fonction de filtrage simple de paquets (stateless inspection) ;
- fonction de filtrage de paquets avec état (stateful inspection) ;
- fonction de filtrage applicatif (ou fare-feu de type Proxy ou proxing applicatif)

☞ Le filtrage simple de paquets :

Le filtrage simple est l'une des premières solutions **Firewall** à avoir été mis en œuvre. Ce système inspecte les paquets IP (en-tête et donnée) des couches réseaux et transport afin d'en extraire l'adresse et le port source et l'adresse et le port de destination et le protocole utilisée. Cette solution permet de déterminer la nature du service demandé et de définir si le paquet IP doit être accepté ou rejeté en fonction des règles définies. **[R 2.2]**

Table I.2.1 : Exemples de règles de Pare-feu : [MFE 2.2]

Règle	Action	IP source	IP destination	Protocole	Port source	Port destination
1	Accept	192.168.10.20	194.154.192.3	TCP	Any	25
2	Accept	Any	192.168.10.3	TCP	Any	80
3	Accept	192.168.10.0/ 24	Any	TCP	Any	80
4	Deny	Any	Any	Any	Any	Any

Les ports reconnus (dont le numéro est compris entre 0 et 1023) sont associés à des services courants (les ports 25 et 110 sont par exemple associés au courrier électronique, et le port 80 au Web). La plupart des dispositifs pare-feu sont au minimum configurés de manière à filtrer les communications selon le port utilisé. Il est généralement conseillé de bloquer tous les ports qui ne sont pas indispensables (selon la politique de sécurité retenue).

Le port 23 est par exemple souvent bloqué par défaut par les dispositifs pare-feu car il correspond au protocole Telnet, permettant d'émuler un accès par terminal à une machine distante de manière à pouvoir exécuter des commandes à distance. Les données échangées par Telnet ne sont pas chiffrées, ce qui signifie qu'un individu est susceptible d'écouter le réseau et

de voler les éventuels mots de passe circulant en claire. Les administrateurs lui préfèrent généralement le protocole SSH, réputé sûr et fournissant les mêmes fonctionnalités que Telnet.

☞ Le filtrage dynamique :

Le filtrage simple de paquets ne s'attache qu'à examiner les paquets IP indépendamment les uns des autres, ce qui correspond au niveau 3 du modèle OSI. Or, la plupart des connexions reposent sur le protocole TCP, qui gère la notion de session, afin d'assurer le bon déroulement des échanges. D'autre part, de nombreux services (le FTP par exemple) initient une connexion sur un port statique, mais ouvrent dynamiquement (c'est-à-dire de manière aléatoire) un port afin d'établir une session entre la machine faisant office de serveur et la machine cliente.

Ainsi, il est impossible avec un filtrage simple de paquets de prévoir les ports à laisser passer ou à interdire. Pour y remédier, le système de **filtrage dynamique de paquets** est basé sur l'inspection des couches 3 et 4 du modèle OSI, permettant d'effectuer un suivi des transactions entre le client et le serveur. Le terme anglo-saxon est « **stateful inspection** » ou « *stateful packet filtering* », traduisez « *filtrage de paquets avec état* ».

Un dispositif pare-feu de type « stateful inspection » est ainsi capable d'assurer un suivi des échanges, c'est-à-dire de tenir compte de l'état des anciens paquets pour appliquer les règles de filtrage. De cette manière, à partir du moment où une machine autorisée initie une connexion à une machine située de l'autre côté du pare-feu; l'ensemble des paquets transitant dans le cadre de cette connexion sera implicitement accepté par le pare-feu.

Si le filtrage dynamique est plus performant que le filtrage de paquets basique, il ne protège pas pour autant de l'exploitation des failles applicatives, liées aux vulnérabilités des applications. Or ces vulnérabilités représentent la part la plus importante des risques en termes de sécurité.

☞ Le filtrage applicatif :

Le filtrage applicatif permet de filtrer les communications application par application. Le filtrage applicatif opère donc au niveau 7 (couche application) du modèle OSI, contrairement au filtrage de paquets simple (niveau 4). Le filtrage applicatif suppose donc une connaissance des protocoles utilisés par chaque application.

Un firewall effectuant un filtrage applicatif est appelé généralement « passerelle applicative » (ou « proxy »), car il sert à relier entre deux réseaux en s'interposant et en

effectuant une validation fine du contenu des paquets échangés. Le proxy représente donc un intermédiaire entre les machines du réseau interne et le réseau externe, subissant les attaques à leur place. De plus, le filtrage applicatif permet la destruction des en-têtes précédant le message applicatif, ce qui permet de fournir un niveau de sécurité supplémentaire.

Il s'agit d'un dispositif performant, assurant une bonne protection du réseau, pour peu qu'il soit correctement administré. En contrepartie, une analyse fine des données applicatives requiert une grande puissance de calcul et se traduit donc souvent par un ralentissement des communications, chaque paquet devant être finement analysé.

Par ailleurs, le proxy doit nécessairement être en mesure d'interpréter une vaste gamme de protocoles et de connaître les failles afférentes pour être efficace.

Enfin, un tel système peut potentiellement comporter une vulnérabilité dans la mesure où il interprète les requêtes qui transitent par son biais. Ainsi, il est recommandé de dissocier le pare-feu (dynamique ou non) du proxy, afin de limiter les risques de compromission.

☞ Notion de Pare-feu personnel :

Dans le cas où la zone protégée se limite à l'ordinateur sur lequel le Firewall est installé on parle de **Firewall personnel** (*Pare-feu personnel*).

Ainsi, un Firewall personnel permet de contrôler l'accès au réseau des applications installées sur la machine, et notamment empêcher les attaques du type "cheval de Troie", c'est-à-dire des programmes nuisibles ouvrant une brèche dans le système afin de permettre une prise en main à distance de la machine par un pirate informatique. Le Firewall personnel permet en effet de repérer et d'empêcher l'ouverture non sollicitée de la part d'applications non autorisées à se connecter.

Les limites des Firewalls :

Un système pare-feu n'offre bien évidemment pas une sécurité absolue. Les Firewalls n'offrent une protection que dans la mesure où l'ensemble des communications vers l'extérieur passe systématiquement par leurs intermédiaires et qu'ils sont correctement configurés. Ainsi, les accès au réseau extérieur par contournement du Firewall sont autant des failles de sécurité.

C'est notamment le cas des connexions effectuées à partir du réseau interne à l'aide d'un modem ou de tout moyen de connexion échappant au contrôle du Pare-feu.

De la même manière, l'introduction de supports de stockage provenant de l'extérieur sur des machines internes au réseau ou bien d'ordinateurs portables peut porter fortement préjudice à la politique de sécurité globale.

Enfin, afin de garantir un niveau de protection maximal, il est nécessaire d'administrer le pare-feu et notamment de surveiller son journal d'activité afin d'être en mesure de détecter les tentatives d'intrusion et les anomalies. Par ailleurs, il est recommandé d'effectuer une veille de sécurité (en s'abonnant aux alertes de sécurité des CERT par exemple) afin de modifier le paramétrage de son dispositif en fonction de la publication des alertes.

La mise en place d'un Firewall doit donc se faire en accord avec une véritable politique de sécurité.

2.3.3. Fonctionnement du pare-feu sous Linux : [R2.3]

Dans le paragraphe qui suit nous allons aborder une vue générale sur le fonctionnement du Pare-feu Linux.

Le but du Pare-feu est de réceptionner, filtrer et envoyer des trames IP. On pourra filtrer les protocoles TCP/IP, UDP/IP et ICMP. Avec ce Pare-feu, nous pourrons protéger notre LAN de vilains qui rôdent sur l'Internet...

Tout d'abord, nous allons définir deux zones :

- la zone "OUTSIDE" qui est la zone non sûre (là où rôdent les vilains pirates).
- la zone "INSIDE" qui est la zone sûre (notre LAN).

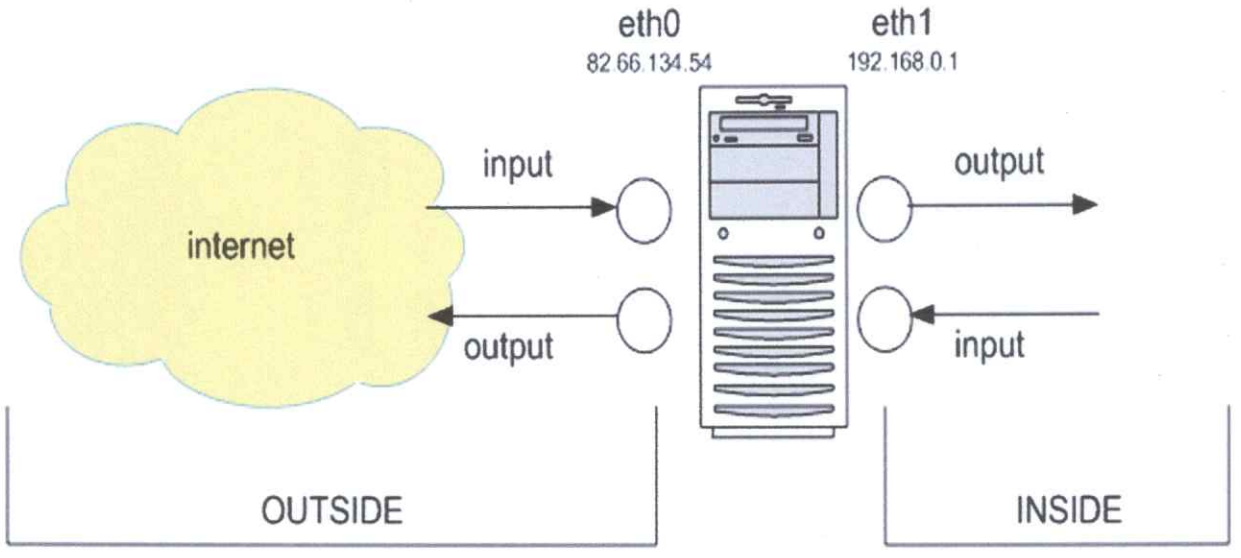


Figure I.2.2 : Les deux interfaces d'un Pare feu.

Le noyau Linux possède des fonctionnalités de filtrage des trames. Une trame arrive, que ce soit par l'une ou l'autre des interfaces Ethernet, en fonction de l'adresse source et de l'adresse distination, le Pare-feu va regarder si l'on a écrit une règle particulière pour cette trame et va l'appliquer, dans le cas contraire une règle par défaut est appliquée.

Pour stocker les règles de filtrage Linux utilise ce que l'on appelle des "chaînes", l'ensemble des chaînes est regroupées dans ce que l'on appelle des "tables".

L'administrateur peut ajouter et retirer des chaînes personnelles dans n'importe quelle table. Au départ, toutes les chaînes sont vides et leur politique par défaut est d'accepter les paquets.

Pour rappel, un paquet IP contient les informations nécessaires au routage du paquet, c'est à dire **adresse IP source** et **adresse IP destination**. Sur cette couche IP, on construit une couche session qui peut être TCP ou UDP. Cette couche apporte d'autres informations comme le **port source** et le **port destination**.

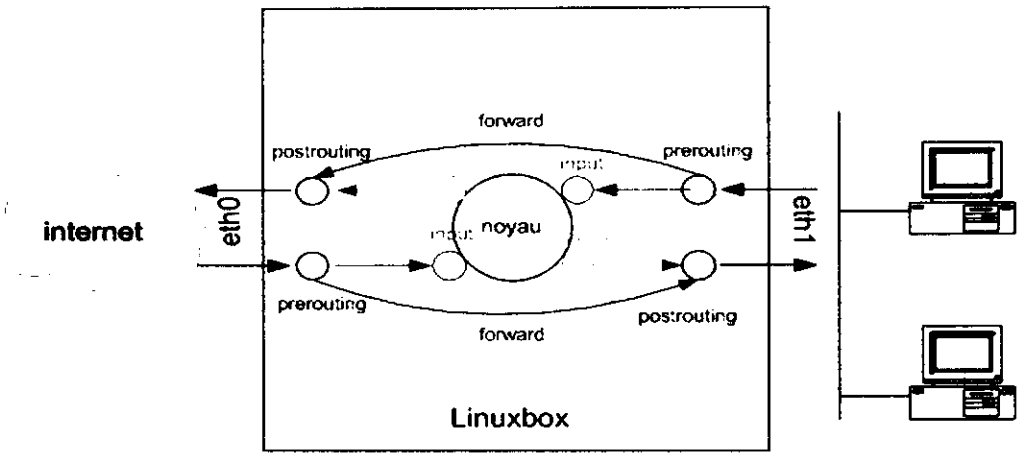


Figure I.2.3 : La structure des différentes chaînes du Pare-feu Netfilter.

2.4. Conclusion :

Il existe aujourd'hui très peut de systèmes sûrs en dehors de l'industrie militaire ou de quelque domaines spécialisées, la sécurité joue un rôle très particulier, parce que la moindre défaillance peut compromettre le bon fonctionnement du système.

Dans ce chapitre nous avons identifiées les différentes fonctions de contrôle d'accès qui peuvent être implémentées par un Firewall, notons que chaque fonction de contrôle d'accès à ces propres avantages et inconvenionts. Dans le prochain chapitre, nous allons aborder une étude plus approfondie sur le **Firewall Netfilter**.

Chapitre 3

Etude du Firewall Netfilter

Dans ce chapitre :

- Introduction
- Fonctionnement général de Netfilter
- Description des commandes IPtables
- Conclusion

3.1. Introduction:

IPtables est un utilitaire Linux (exclusivement) utilisée pour gérer le Pare-feu qui est intégré au noyau Linux 2.4 (et supérieur). L'architecture du noyau pour le système de Pare-feu s'appelle '**Netfilter**'.

Un pare-feu est un système qui filtre les connections réseau. Cela permet d'interdire des connections sur certains ports d'un ordinateur, de limiter le nombre de connexions, de limiter les bandes passantes …de gérer de nombreux aspects des flux réseau.

Netfilter à de nombreuses fonctionnalités, nous en étudierons les parties essentielles (filtrage de paquets, suivi de connections, NAT …).

3.2. Fonctionnement général de Netfilter : [R3.1]

Le principe est simple, lorsque la carte réseau reçoit un paquet celui-ci est transmis à la partie Netfilter du noyau, Netfilter va ensuite étudier ce paquet (les entêtes et le contenu) et en ce basant sur des règles que l'administrateur aura défini, il va choisir de laisser passer le paquet intact, de modifier ce paquet, de le transmettre à une autre machine ou encore d'interdire le passage.

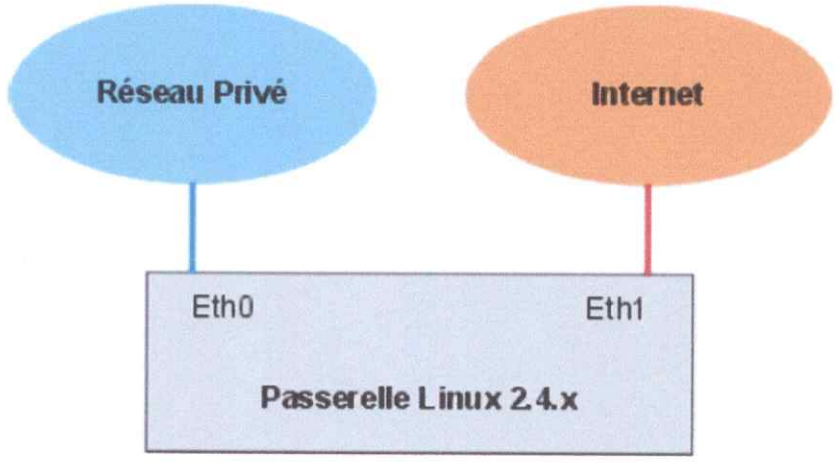


Figure I.3.1 : Topologie de la machine Linux.

Netfilter fonctionne en utilisant trois tables invariantes : **Filter, Nat** et **Mangle**, chaque paquet passe systématiquement par une ou plusieurs *tables*. A l'intérieur de ces tables, les paquets parcourent des *chaînes* qui contiennent les règles de traitement.

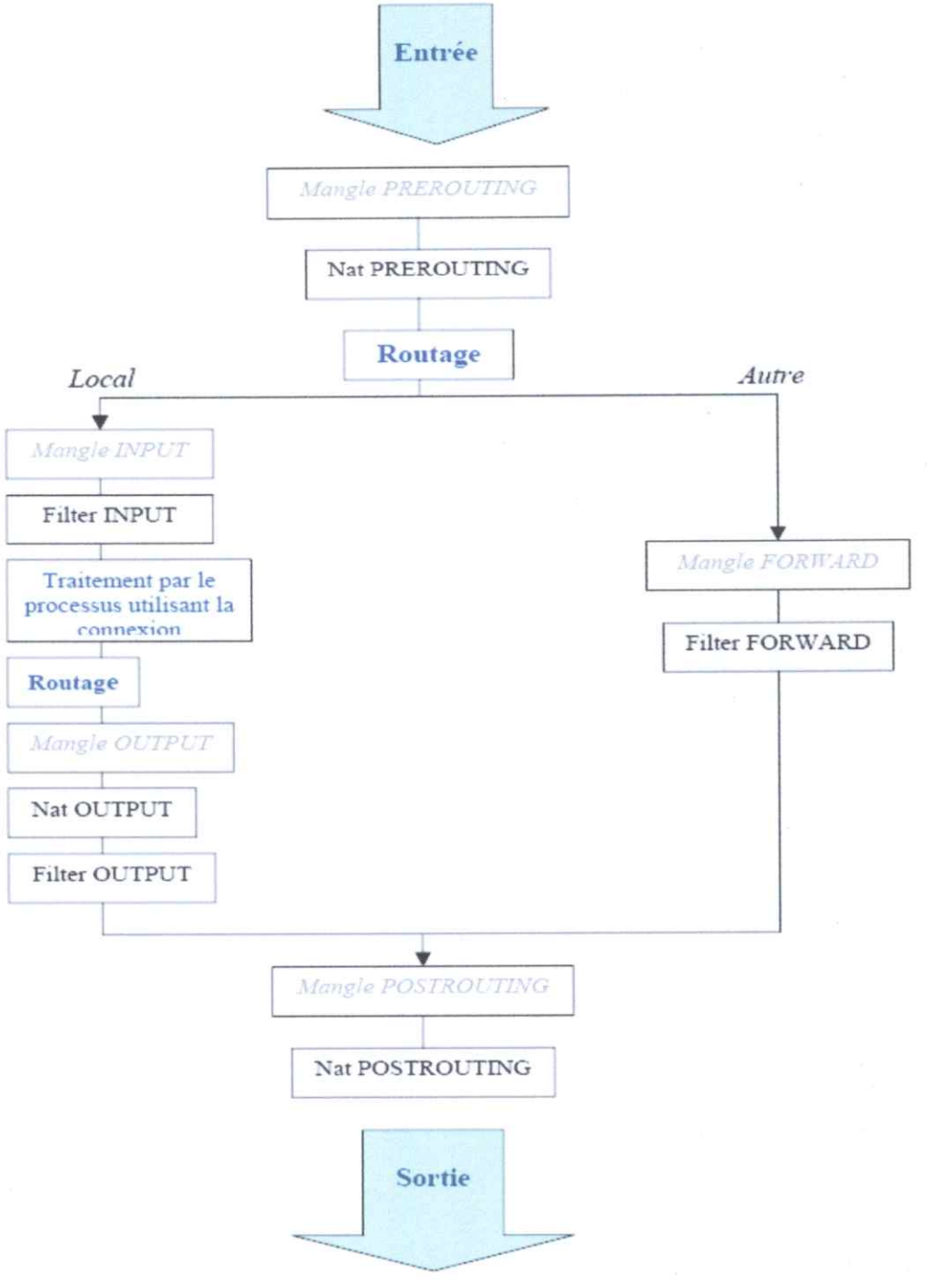


Figure I.3.2 :L'acheminement des paquets IP à travers les différentes chaînes de Netfilter.

Netfilter permet à un administrateur de définir comment Linux doit traiter les paquets réseau entrants et sortants (manipulation des règles). L'administrateur peut ajouter et retirer des chaînes personnelles dans n'importe quelle table, créer de nouvelles chaînes.....

Chaque règle spécifie des critères de sélection (adresse source, protocole, etc.) et ce qui doit advenir des paquets correspondants (rejet, autorisation, archivage, etc.). Les critères disponibles et les actions possibles ne sont pas toujours les mêmes suivant la chaîne dans laquelle on ajoute la règle. Suivant l'action choisie, le paquet parcourt ou non les règles suivantes de la chaîne. Si aucune règle ne correspond au paquet, c'est la politique par défaut de la chaîne qui décide du sort du paquet.

Tables

Netfilter est fourni avec trois tables principales, mais des modules d'extension peuvent en créer de nouvelles. Chaque table contient des chaînes prédéfinies à travers lesquelles les paquets vont passer. L'administrateur peut ajouter et retirer des chaînes personnelles dans n'importe quelle table.

- Table Filter :

Cette table est responsable du filtrage (bloquer ou autoriser le passage d'un paquet). Tous les paquets passent à travers cette table, dans l'une des chaînes suivantes :

- *Chaîne INPUT* — Tous les paquets **destinés à** cet ordinateur passent dans cette chaîne.
- *Chaîne OUTPUT* — Tous les paquets **crées par** cet ordinateur passent dans cette chaîne.
- *Chaîne FORWARD* — Tous les paquets **traversant** cet ordinateur passent dans cette chaîne (dans le cas d'un routeur, par exemple).

- Table Nat :

Cette table est responsable de la translation d'adresses et de ports. Seul le premier paquet de chaque connexion passe à travers cette table. Les règles vont alors déterminer comment seront modifiés tous les paquets relatifs à cette connexion.

- *Chaîne PREROUTING* — Les paquets arrivant de l'extérieur passent dans cette chaîne avant d'être routés. Elle permet de modifier la **destination** de la connexion pour, par exemple, rediriger une connexion.

- *Chaîne POSTROUTING* — Les paquets passent dans cette chaîne après la décision de routage, juste avant que le paquet soit expédié. Elle permet la modification de la **source** de la connexion. On peut ainsi camoufler la véritable origine d'une connexion.
- *Chaîne OUTPUT* — Elle fonctionne comme la chaîne PREROUTING mais pour les connexions issues d'un processus de l'ordinateur.

- Table mangle :

Cette table est responsable de la **transformation** des options des paquets, comme la qualité de service. Tous les paquets passent à travers cette table, dans une ou plusieurs des chaînes suivantes :

- *Chaîne PREROUTING* — Tous les paquets **entrant** cet ordinateur passent dans cette chaîne, avant le routage.
- *Chaîne INPUT* — Tous les paquets **destinés** à cet ordinateur passent dans cette chaîne.
- *Chaîne OUTPUT* — Tous les paquets **créés par** cet ordinateur passent dans cette chaîne.
- *Chaîne FORWARD* — Tous les paquets **traversant** cet ordinateur passent dans cette chaîne.
- *Chaîne POSTROUTING* — Tous les paquets **quittant** cet ordinateur passent dans cette chaîne, après le routage.

3.3. Description des commandes IPtables : [R3.2]

La syntaxe générale pour l'écriture d'une règle IPtables est la suivante:

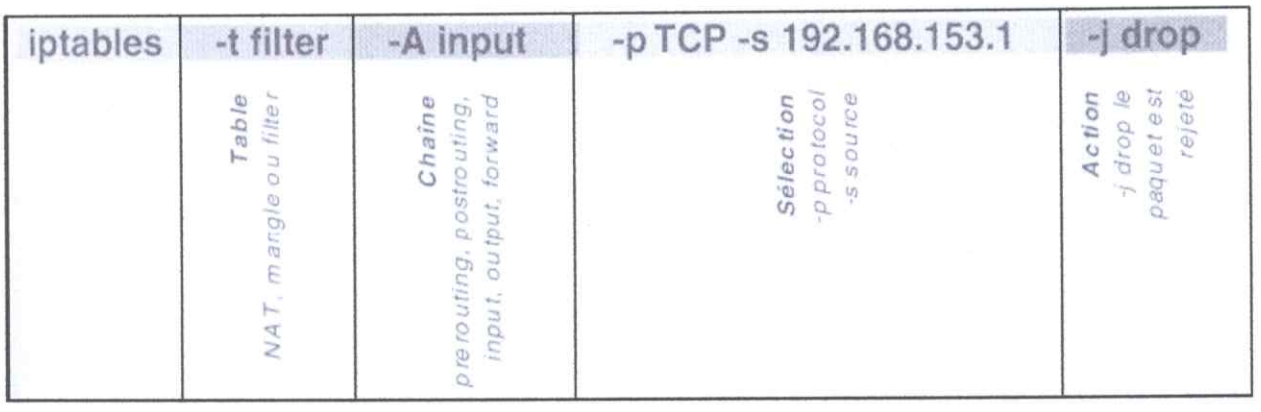


Figure I.3.3 : La syntaxe générale des commandes IPtables.

Par défaut il y a trois chaînes INPUT, OUTPUT et FORWARD que tu ne peux pas effacer.

- Regardons les opérations pour administrer les chaînes :

1. Créer une nouvelle chaîne (-N).
2. Effacer une chaîne vide (-X).
3. Changer la règle par défaut pour une chaîne de départ (-P).
4. Lister les règles dans une chaîne (-L).
5. Retirer les règles d'une chaîne (-F).
6. Mettre à zéro les compteurs de bits et de paquets d'une chaîne (-Z).

- Il y a plusieurs manières de manipuler une règle dans une chaîne :

1. Ajouter une nouvelle règle à la chaîne (-A).
2. Insérer une nouvelle règle à une position dans la chaîne (-I).
3. Remplacer une règle à une position dans la chaîne (-R).
4. Supprimer une règle à une position dans la chaîne (-D).
5. Supprimer la première règle qui convient dans une chaîne (-D).

- Liste des actions possibles:

Table I.3.1 : Les actions possibles des règles Iptables :

-j ACCEPT	Le paquet est accepté.
-j DROP	Le paquet est rejeté.
-j REJECT	Le paquet est rejeté, l'expéditeur est averti de l'indisponibilité du service.
-j QUEUE	Le paquet est envoyé à une application.
-j LOG	Le paquet est envoyé au système « syslog ».
-j MARK	Le paquet est marqué.
-j TOS	Modifie le « Type Of Service » du paquet.
-j MIRROR	Renvoie le paquet à l'expéditeur.
-j SNAT	L'adresse source du paquet est translatée.
-j DNAT	L'adresse destination du paquet est translatée.
-j MASQUERADE	L'adresse source du paquet est translatée.
-j REDIRECT	R-direction d'un port vers un autre.

- Les Paramètre d'une fonction peuvent être:

-i interface: appliquer la règle sur cette interface d'entrée

-o interface: appliquer la règle sur cette interface de sortie

-t table: table concernée.

-s [!] X.X.X.X/lg: adresse source, longueur du masque

Le ! Signifie la négation.

-d [!] X.X.X.X/lg : adresse destination (et longueur du masque).

-p protocole: tcp / udp / icmp / all.

--source-port [!] [Port [: port]] : port source (ou intervalle de ports).

--destination port [!] [port [: port]] :port destination (ou intervalle de ports).

--tcp-flags [!] masq comp flags de paquets tcp (ex. SYN).

-j cible : ACCEPT/DROP/QUEUE/RETURN/REDIRECT/MASQUERADE / DNAT/ SNAT/ LOG.

Exemple de règle :

- Dans cet exemple on interdit sur l'interface eth1 tout ce qui ne vient pas du réseau 195.220.162.0/24.

```
iptables -F INPUT [-t filter]
iptables -P INPUT ACCEPT
iptables -A INPUT -i eth1 -s ! 195.220.162.0/24 -j DROP
iptables -A FORWARD -i eth1 -s ! 195.220.162.0/24 -j DROP
```

Les tables de Netfilter sont stockées en mémoire et sont effacées lors des rechargements.

- Créer une nouvelle chaîne*:*

iptables -N votre_nom_de_chaîne

- Ajouter une nouvelle règle à une chaîne : iptables -A

Pour l'exemple, nous "droperons" tous les paquets icmp en provenance de 127.0.0.1:
iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP

- Supprimer les règles d'une chaîne

iptables -F nom_de_votre_chaîne
Attention! Si vous ne spécifiez aucune chaîne, toutes les chaînes seront vidées.

- Effacer une chaîne:

iptables -X ma_chaîne

- Les polices de sécurité de chaînes

Refuser tout ce qui provient de l'hôte local en entrée.
iptables -A INPUT -s 127.0.0.1 -j DROP

- Utiliser le complément avec !

iptables -A INPUT -s ! 127.0.0.1 -j DROP

- Mentionner un protocole avec -p

iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP

- Mentionner les flags --tcp_flags

iptables -A INPUT -p tcp --tcp-flags ALL -j DROP

3.4. Conclusion:

Étant le Pare-feu officiel de Linux, Netfilter est très répandu et bien documenté. Il est facile d'obtenir de l'aide sur les forums, des interfaces graphiques existent pour simplifier sa configuration, l'architecture modulaire permet une grande évolution et l'adaptation à des besoins spécifiques…..

Mais dans la contre partie Netfilter ne gère pas certaines technologies comme IPsec, les proxys, les IDS ou les serveurs d'authentification, en plus de ça chaque appel à la commande IPtables accède à la configuration de Netfilter dans le noyau, pour créer une grosse configuration, il faut souvent des milliers d'appels à IPtables!?.

Chapitre 4

Etude des DMZs (Demilitarized zone)

Dans ce chapitre :

- Introduction
- Définitions
- Firewall et DMZ ?
- Conclusion

4.1. Introduction:

Lorsque certaines machines du réseau interne ont besoin d'être accessibles depuis l'extérieur (comme c'est le cas par exemple pour un serveur Web, un serveur de messagerie, un serveur FTP public, ...), il est souvent nécessaire de créer une nouvelle interface vers un réseau à part, accessible aussi bien du réseau interne que de l'extérieur, sans pour autant risquer de compromettre la sécurité de l'entreprise (c'est à dire du réseau local).

On parle ainsi de zone démilitarisée (souvent notée **DMZ** pour DeMilitarized Zone), qui désigne une zone isolée hébergeant des applications mises à disposition du public, comme il est montré dans la figure I.4.1 **[RI.4.2]**

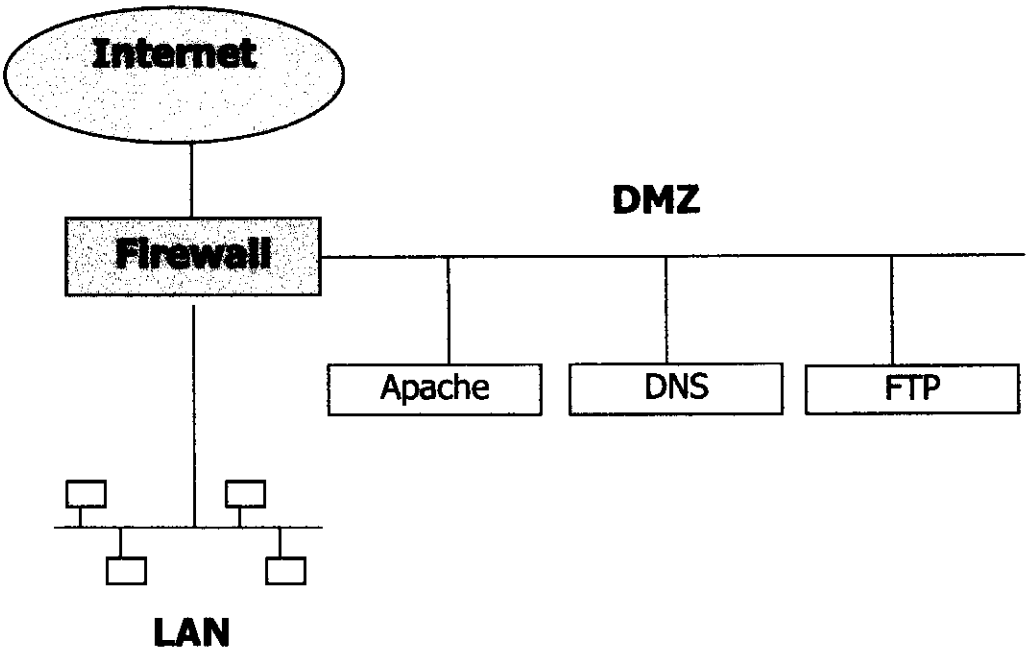


Figure I.4.1 : Architecture d'un réseau d'entreprise disposant d'une DMZ isolée par un Firewall.

4.2. Définitions :

Puisque une DMZ est un concept, il existe plusieurs définitions, on cite quelque unes :

- Une zone démilitarisée dit DMZ est un segment du réseau sur lequel des ressources internes sont publiées pour des clients externes.

Exemple : Serveur Web hébergeant un site Internet d'une entreprise. Le serveur Web est alors publié pour le public externe dans la **zone démilitarisée.**

Un serveur Web hébergeant un site Intranet serait par contre sur un segment du réseau privé.

- C'est une zone située à l'interface entre Internet et un réseau local où sont placés les serveurs de sites Web, de messagerie..., cette configuration permet d'isoler les serveurs et ainsi de protéger le réseau local tout en économisant la bande passante de celui-ci.
- DMZ (zone démilitarisée) est une utilisation particulière d'un ou deux Firewalls hardware (pare feu). Ceci permet de connecter un serveur sur deux réseaux à la fois mais de bloquer l'accès entre les deux réseaux. L'ordinateur est typiquement un serveur Internet partagé entre un réseau local et Internet. La DMZ empêche l'accès des visiteurs Internet vers le réseau interne.
- Sur un pare-feu, on appelle DMZ une zone qui n'est ni publique, ni interne. Ces zones ne peuvent exister que si le pare-feu possède plus de deux interfaces réseau, une pour la connexion au réseau externe et l'autre pour la connexion au réseau interne.

Chaque une de ces définitions décrit d'une manière globale les différentes architectures existantes pour la mise en place d'une politique de sécurité en DMZ :

- La DMZ est une zone intermédiaire entre le réseau interne et le réseau externe (tout flux entre les deux réseaux doit passer par l'intermédiaire de la DMZ).
  Deux Firewalls sont nécissaires pour l'implémentation de cette architecture, le premier gère le flux entre la DMZ et le réseau externe, le deuxième gère le flux entre la DMZ et le réseau interne.
- La DMZ est une zone isolée hébergeant des applications mises à disposition du public, dans ce cas un ou deux Firewalls sont nécissaires.

Dans ce qui suit nous détaillons le deuxième cas (La DMZ est une zone isolée hébergeant des applications mises à disposition du public) avec l'utilisation d'un seul Firewall.

4.3. Firewall et DMZ ? : [RI.4.1]

Si un pare-feu possède trois interfaces au moins (ou trois "pattes"), il est possible de créer une DMZ sur la troisième interface. On y placera les serveurs qui ont besoin de sortir sur l'extérieur, mais qui ont également besoin d'être protéger des menaces internes. Ainsi, si des utilisateurs en interne veulent faire des opérations frauduleuses sur les serveurs, ils devront franchir la barrière du pare-feu et les règles de filtrage mises en place.

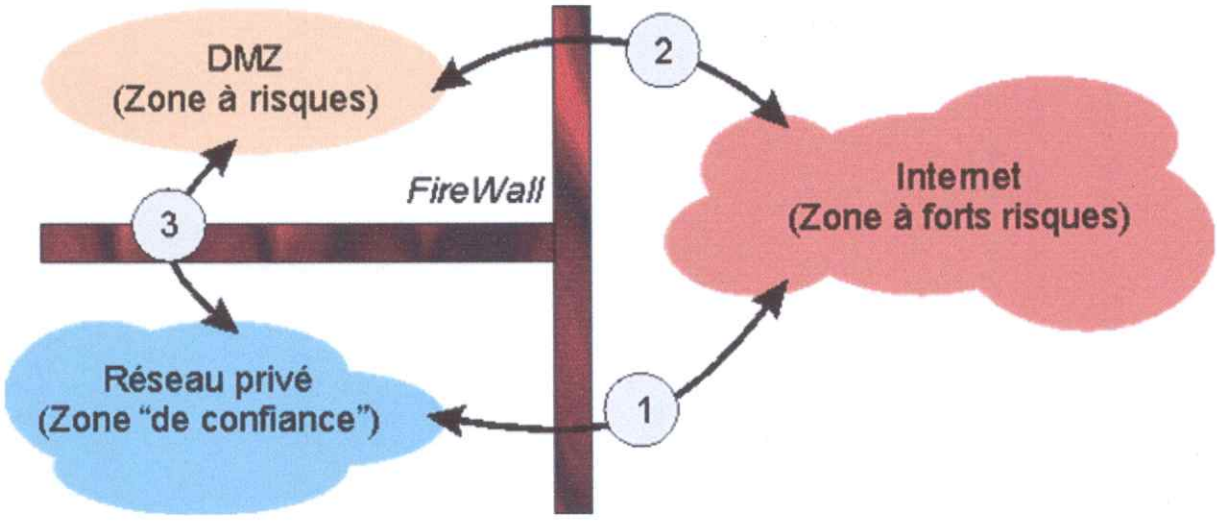


Figure I.4.2 : Une DMZ protégée par le Firewall.

Typiquement, une installation complète contient :

- Un **réseau privé**, dont on considère qu'il ne sera pas utilisé pour attaquer notre système informatique. Dans cette zone, il n'y a que des clients du réseau et des serveurs qui sont inaccessibles depuis l'Internet. Normalement, aucune connexion, au sens TCP du terme, aucun échange, au sens UDP du terme, ne peuvent être initiés depuis le Net vers cette zone.
- Une "**DMZ**" (Zone Démilitarisée), qui contient des serveurs accessibles depuis le Net et depuis le réseau privé. Comme ils sont accessibles depuis le Net, ils risquent des attaques. Ceci induit deux conséquences :
  - Il faut étroitement contrôler ce que l'on peut faire dessus depuis le Net, pour éviter qu'ils se fassent "casser" trop facilement,
  - Il faut s'assurer qu'ils ne peuvent pas accéder aux serveurs de la zone privée, de manière à ce que si un pirate arrivait à en prendre possession, il ne puisse directement accéder au reste du réseau.

Les trois types de communications marquées **1**,**2** et **3** sur l'illustration seront donc soumis à des règles de passage différentes. Le dispositif qui va permettre d'établir ces règles de passages est le **Firewall**. Techniquement, ce pourra être un logiciel de contrôle installé sur un routeur.

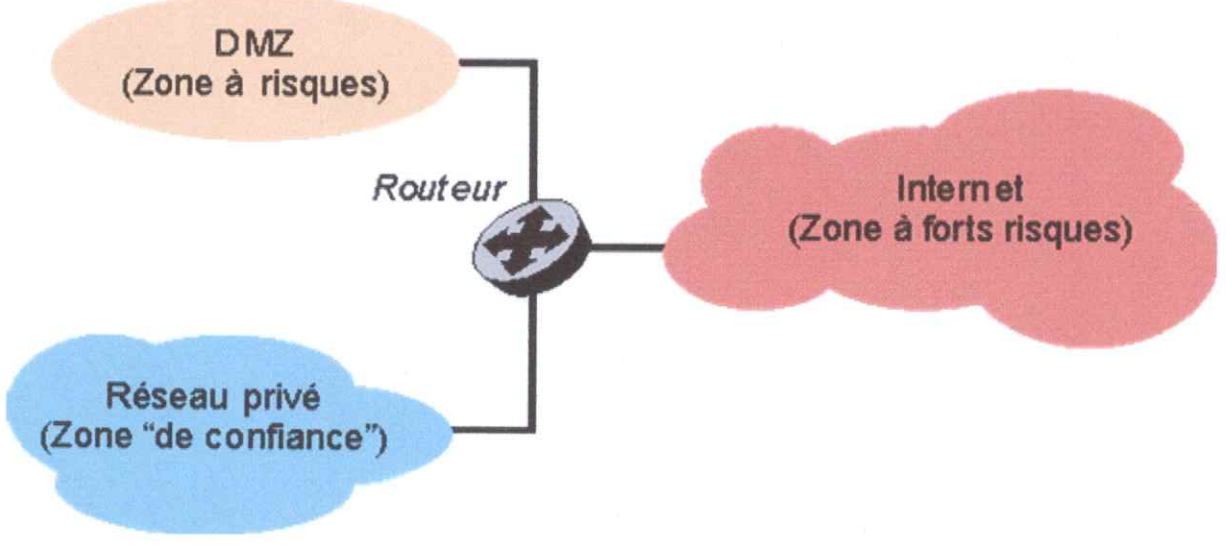


Figure I.4.3 : Une DMZ protégée par un routeur sur le quel est installée un logiciel de contrôle.

Nous aurons, par exemple :

- Le réseau privé dans une classe d'adresses IP privées, comme 192.168.0.0,
- la DMZ dans une autre classe privée comme 192.168.1.0,
- une adresse publique attribuée au routeur sur le Net.

Le routeur cumule ici deux fonctions :

- Le **routage** proprement dit, pour permettre aux paquets de passer d'une zone à l'autre alors que ces zones ne sont pas situées dans le même réseau IP,
- Le **filtrage** du trafic entre les diverses zones, c'est la fonction du Firewall.

Bien entendu, qui peut le plus peut le moins, il peut ne pas y avoir de DMZ, si l'on n'a pas besoin d'exposer des serveurs sur le Net.

Sur des petites structures, la DMZ peut être implantée sur le routeur (ou le Firewall) lui-même. En effet, un routeur peut très bien n'être rien d'autre qu'un serveur avec plusieurs interfaces réseau et un logiciel de routage. Ce n'est bien entendu pas une solution très "propre" au sens de la sécurité.

4.4. Description du flux échangée entre les différentes entités:

- **Entre le réseau privé et le Net :**

Toujours typiquement, ce sont les clients du réseau (les utilisateurs) à qui l'on va donner des possibilités d'accéder au Net, comme par exemple le surf ou la messagerie. Toutes les requêtes partent du réseau privé vers le Net. Seules les réponses à ces requêtes doivent entrer dans cette zone. Les accès peuvent être complètement bridés (les clients du réseau privé n'ont aucun droit d'accès vers le Net, ça nuit à leurs productivité. Seul le patron y a droit). Où alors, les utilisateurs ne pourront consulter qu'un nombre de sites limités, dans le cadre de leurs activités professionnelles exclusivement. Très généralement, cette zone est construite sur une classe d'adresses privées et nécessite donc une translation d'adresse pour accéder au Net. C'est le routeur (Firewall) qui se chargera de cette translation.

- **Entre la DMZ et le Net :**

Ici, nous avons des serveurs qui doivent être accessibles depuis le Net. Un serveur Web, un serveur de messagerie, un FTP.... Il faudra donc permettre de laisser passer des connexions initiées depuis l'extérieur. Bien entendu, ça présente des dangers, il faudra surveiller étroitement et ne laisser passer que le strict nécessaire.

Si l'on dispose d'adresses IP publiques, le routeur (Firewall) fera un simple routage. Si l'on n'en dispose pas, il devra faire du "**port forwarding**" pour permettre, avec la seule IP publique dont on dispose, d'accéder aux autres serveurs de la DMZ. Cette technique fonctionne bien sur un petit nombre de serveurs, mais devient très vite un casse-tête si, par exemple, plusieurs serveurs HTTP sont présents dans la DMZ.

- **Entre le réseau privé et la DMZ :**

Les accès devraient être à peu près du même type qu'entre la zone privée et le Net, avec un peu plus de souplesse. En effet, il faudra :

- Mettre à jour les serveurs web,
- Envoyer et recevoir les messages, puisque le SMTP est dedans
- Mettre à jour le contenu du FTP (droits en écriture).

En revanche, depuis la DMZ, il ne devrait y avoir aucune raison pour qu'une connexion soit initiée vers la zone privée.

4.5. Conclusion:

La mise en place d'une politique de sécurité en DMZ doit prendre en considération les besoins de l'entreprise en se basant sur les facteurs suivants : la différence entre les niveaux de sécurité des sous réseaux (la politique d'échange de données entre ces derniers), la taille du réseau, l'emplacement du Firewall, le risque d'être visible depuis l'extérieur ..., ce qui prouve l'existence de plusieurs architectures de sécurité en DMZ.

Partie II

Conception

Chapitre 1

Proposition d'un formalisme pour la modélisation des DMZs

Dans ce chapitre :

- ❖ Problématique
- ❖ Solution proposée
- ❖ L'environnement de développement
- ❖ L'analyse lexicale avec Lex
- ❖ L'analyse grammaticale avec Yacc

Notre travail est la réalisation d'un outil permettant la conception et l'implémentation des zones démilitarisées (DMZ) de manière à faciliter la configuration du Pare-feu sous linux via la commande IPtables. L'objectif de ce chapitre est de représenter l'architecture de la DMZ proposée.

1.1. Problématique:

La politique économique grandissante de la société **NAFTAL** la met en possession d'un réseau informatique vaste liée à l'Internet. Le besoin de l'entreprise à publier ses ressources internes pour ses clients externes lui impose le choix de la mise en place de la politique de sécurité en DMZ.

Vu que le système d'exploitation utilisé dans les différents services informatiques de l'entreprise est Linux, la mise en place d'une politique de sécurité est faite par la configuration du Firewall Netfilter via la commande IPtables pour la mise en œuvre de la DMZ. Le problème rencontré est que l'administrateur du réseau trouve des défficultés à administrer Netfilter dès que le nombre de règles de filtrage augmente.

1.2. Solution proposée :

Notre objectif est donc de réaliser un outil d'aide à la conception et l'implémentation des zones démilitarisées. Ce dernier génère automatiquement la liste des commandes IPtables correspondantes au programme source saisie par l'utilisateur.

La solution qu'on a proposée est basée sur le schéma de la figure **II.1.1**, donnant une vue sur l'architecture de sécurité en DMZ proposée pour le réseau informatique de l'entreprise. Cette dernière assure une configuration simple et sûre du Firewall et prend en considération les besoins de l'entreprise.

La figure **II.1.1** ci-après, montre l'existence de trois instances de l'entité zone, en plus d'une entité Firewall :

- La première instance de l'entité zone est celle du réseau privé (LAN), on considère qu'elle ne sera pas utilisée pour attaquer notre système informatique. Dans cette zone, il n'y a que des clients du réseau et des serveurs qui sont inaccessibles depuis la zone du Net et les DMZs.
- La deuxième instance est celle représentée par les DMZs: elle contient des serveurs accessibles depuis le Net dans le cas d'une demande de service, et depuis le réseau privé dans le cas d'une mise à jour.
- La troisième instance est celle représentée par le Net ou réseau extérieur, c'est la zone à risques.

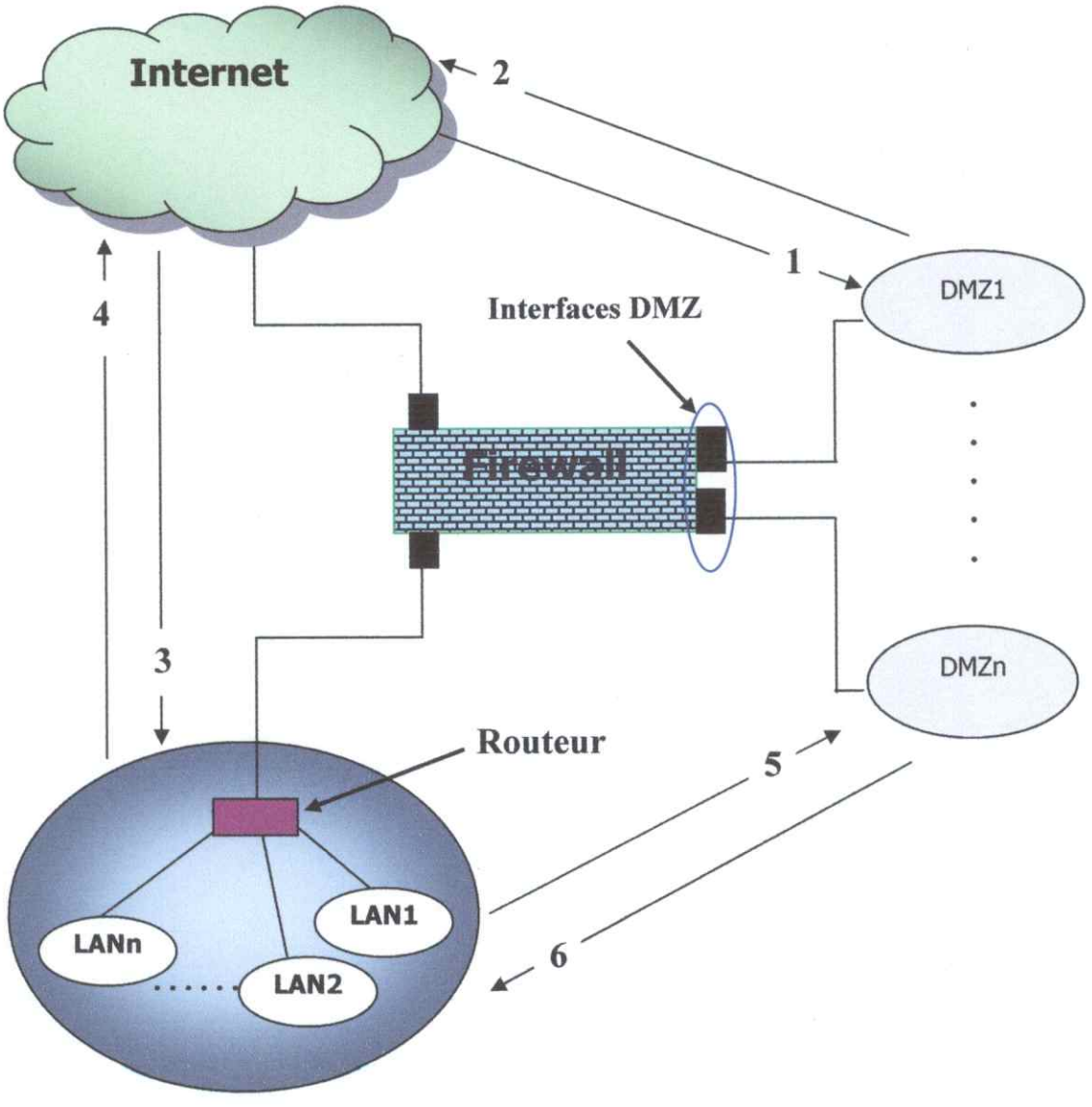


Figure II.1.1 : Le schéma de DMZ considéré et les différents flux échangés.

Les deux premières zones sont liées aux interfaces privées du Firewall, tant dit que la zone du Net doit être liée à l'interface publique.

Entre les trois instances de zones définies, la figure **II.1.1** montre l'existence de six chemins que les flux de paquets IP suivent pour le passage d'une zone à l'autre sous le contrôle du Firewall. Ce dernier applique pour chaque paquet entrant ou sortant deux fonctionnalités de filtrage et/ou routage (tout dépend de la nature du paquet).

Note : il faut préciser que la politique par défaut du Firewall qu'on a choisi est DROP (refuser le passage de tout paquet pour lequel aucune règle n'est applicable).

Le sens N° 1 (Internet - DMZ) :

La DMZ héberge des services accessibles depuis le net. Le Firewall doit laisser passer toute connexion autorisée, initiée depuis le Net. Il doit assurer une opération de translation d'adresse et/ou de port (c'est l'opération de publication).

Le sens N° 2 (DMZ - Internet):

Dans ce sens, il ne faut laisser passer que les flux de réponse aux différentes requêtes initiées depuis un poste du Net vers un poste serveur de la DMZ.

Le sens N° 3 (Internet - LAN):

Dans ce sens, il ne faut laisser passer que les flux de réponse aux différentes requêtes initiées depuis un poste du LAN vers un poste serveur du Net.

Le sens N° 4 (LAN - Internet) :

Dans ce sens, l'administrateur précise l'ensemble des postes du LAN à qui il va attribuer le droit de naviguer sur le Net. En plus de l'opération de filtrage, le Firewall doit assurer la translation des adresses privées du LAN.

Le sens N° 5 (LAN - DMZ) :

Les paquets IP traversent ce sens dans le cas ou l'administrateur veut par exemple

- permettre une mise à jour des serveurs web à partir d'un poste du LAN;

- Envoyer et recevoir les messages puisque le SMTP est dedans;
- Mettre à jour le contenu du FTP. Dans ce cas, le Firewall joue le rôle d'un filtre de paquets IP.

Le sens N° 6 (DMZ - LAN) :

Ce sens est strictement interdit sauf s'il s'agit d'un flux de repense a une requête initiée par un poste du LAN. Il n'y a aucune raison pour qu'une connexion soit initiée d'un poste de la DMZ vers un poste de la zone privée.

1.3. L'environnement de développement :

- La réalisation de notre logiciel nécessite comme matériel un PC à une configuration suffisante pour l'installation de Linux et les divers serveurs des services réseaux.
- **- Le système d'exploitation :**

  Le système d'exploitation utilisé pour le développement de notre application est Linux **Fedora core 4**, avec le noyau 2.2.0.9. Ce dernier implémente automatiquement le module Netfilter et les deux outils Lex (générateur d'analyseur lexicale) et Yacc (**Y**et **A**nother **C**ompiler **C**ompiler : générateur d'analyseur syntaxique).

  - Le langage de programmation :

  Notre choix s'est porté sur le langage C, langage cible des compilateurs Lex et Yacc utilisées pour le développement de notre application.

  Lex et Yacc sont des outils très utiles pour toutes les personnes ayant à manipuler une grammaire. En particulier à ceux qui écrivent des compilateurs ou des interpréteurs.

1.4. L'analyse lexicale avec Lex:

Lex est un générateur d'analyseur lexical basé sur les expressions régulières. Le programme Lex vous permet de créer un programme qui, pour chaque expression régulière retrouvée, pourra exécuter des commandes que vous aurez prédéfinies (écrites en C). Vous pouvez par exemple, très facilement, faire un programme qui reprend un texte et mets en minuscules tous les mots qui sont en majuscule. Ou bien de mettre en majuscule toutes les premières lettres d'un mot qui est précédé d'un point.

Le but de l'analyse lexicale est de transformer une suite de symboles en **terminaux** (un terminal peut être par exemple un nombre, un signe "+", un identificateur, etc.) Une fois cette transformation effectuée, la main est passée à l'analyseur syntaxique.

1.5. L'analyse grammaticale avec Yacc

'Lex' est un générateur d'analyseur lexical, il ne peut pas effectuer une analyse grammaticale. C'est-à-dire que 'Lex' ne vous permet pas de saisir une grammaire complexe d'un langage et de l'analyser. C'est le rôle de **'Yacc'**. 'Yacc' vous permet de faire ce que l'on appelle un 'parser'.

'Yacc' est un analyseur grammatical qui a été spécialement conçu pour utiliser 'Lex' comme analyseur lexical. Ils marchent ensemble, chacun réalisant une tâche à son niveau. La partie 'Lex' lit le fichier, décode les tokens et les renvoie à la partie 'Yacc'. La partie 'Yacc' ne lit pas le fichier en entrée mais seulement les tokens envoyés par 'Lex' au travers de la fonction « yylex (). » La communication entre les deux se passe au niveau des tokens et aussi de la variable « yylval » (Voire. Annexe A).

Chapitre 2

Conception du préprocesseur à réaliser

Dans ce chapitre :

- Introduction
- Description du langage du préprocesseur à réaliser
- Conclusion

2.1. Introduction :

La réalisation du préprocesseur proposé nécessite le passage par deux étapes fondamentales : la première étape est la conception : c'est la proposition et la description du langage d'entrée (langage source). La deuxième étape est l'implémentation du préprocesseur proprement dite.

2.2. Description du langage du préprocesseur à réaliser :

Le programme qu'on va implémenter doit respecter d'une manière stricte toutes les règles décrites par la grammaire suivante que nous avons mise au point. Cette grammaire s'inspire de l'usage des commandes IPtables en vue de configurer le Firewall Netfilter.

(1) lang ---> DECLARATION declar CONFIGURATION conf

Un programme écrit dans notre langage doit être structuré en deux sections :

- La section de déclarations : qui doit commencer par le mot clé « **DECLARATION** ».
- La section de configuration : qui doit commencer par le mot clé « **CONFIGURATION** ».

(2) declar ---> fw zone service

La section de déclarations doit comporter la déclaration des trois entités : Firewall, zones et services.

(3) fw --> FW '{' interfacez interfacenet '}'

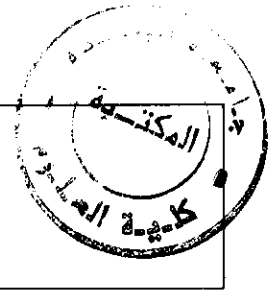
La déclaration de l'entité Firewall, doit commencer par le mot clé « **FW** », suivi de la déclaration des interfaces privées générées par le non terminal <interfacez>, et de l'interface publique générée par le non terminal <interfacenet>.

(4) interfacez ---> interfacez1 interfacez2

<**interfacez**> génère l'interface privée liée au LAN, et <**interfacez1**> génère les interfaces privées liées aux DMZs.

(5) interfacez1 ---> LAN ':' ADRIP ',' IDFIFACE

Le mot clé « **LAN** » est utilisé pour déclarer l'interface liée au LAN caractérisée par une adresse IP et un identifiant reconnue par le noyau Linux (le mot clé « **eth** » suivi d'un constant).



(6) interfacez2 ---> ZONE CONSTANT ':' ADRIP ',' IDFIFACE ifaceplus

(7) ifaceplus ---> interfacez2

(8) | ε

Le mot clé « **ZONE CONSTANT** » est utilisé pour déclarer les interfaces liées aux DMZs. Le non terminal <ifaceplus> est utilisé pour générer plusieurs interfaces. Comme toutes les interfaces du Firewall une interface liée à une DMZ est caractérisée par son adresse IP et un identifiant reconnue par le noyau Linux.

(9) interfacenet ---> NET ':' ADRIP ',' IDFIFACE

Le mot clé « **NET** » est utilisé pour déclarer l'interface publique (liée au Net).

(10) zone ---> ZONES '{' zone1 zone2 '}'

Cette règle permet la déclaration de plusieurs zones qui doit débuter par le mot clé «**ZONES**».

(11) zone1 ---> LAN ':' ADRR

La déclaration de la zone du réseau locale doit commencer par le mot clé « **LAN** » suivi d'une adresse réseau qui est leur attribuée.

(12) zone2 ---> ZONE CONSTANT ':' ADRR suiteplus

(13) suiteplus ---> zone2

(14) | ε

La déclaration de la zone DMZ doit commencer par le mot clé « **ZONE CONSTANT** » suivi d'une adresse réseau qui est leur attribuée.

" **CONSTANT** " est le numéro séquentiel de la zone.

Le non terminal <suiteplus>, permet la déclaration de plusieurs DMZs.

(15) service ---> SERVICES '{' servicez '}'

La déclaration des services, doit commencer par le mot clé « **SERVICE** » suivi de la liste de services hébergés par les DMZs.

(16) servicez ---> IDFS ZONE CONSTANT ':' ADRIP ',' CONSTANT ',' PROTOCOLE serviceplus

Chaque service est caractérisée par :

- Un identificateur < **IDFS** >;
- Un identificateur de la DMZ qui l'héberge < **ZONE CONSTANT** > ;
- Une adresse du poste et un numéro de port < **CONSTANT** >.
- Sa qualité (fiable ou bien non fiable).

(17) serviceplus ---> servicez

(18) | ε

Le non terminal <serviceplus>, permet une génération de plusieurs services.

(19) conf ---> confzone confplus

(20) confplus ---> conf

(21) | ε

Ces deux dernières règles permettent la génération d'une ou de plusieurs règles de configuration.

(22) confzone : AUTORISER ACCEE ':' LAN '-' '>' PROTOCOLE NET

(23) | AUTORISER ACCEE ':' LAN '-' '>' CONSTANT ',' PROTOCOLE NET

(24) | AUTORISER ACCEE ':' LAN '-' '>' adripv ',' PROTOCOLE NET

(25) | AUTORISER ACCEE ':' LAN '-' '>' adripv ',' CONSTANT ',' PROTOCOLE NET

(26) | AUTORISER ACCEE ':' adripv LAN '-' '>' PROTOCOLE NET

(27) | AUTORISER ACCEE ':' adripv LAN '-' '>' CONSTANT ',' PROTOCOLE NET

(28)	**| AUTORISER ACCEE ':' adripv LAN '-' '>' adripv ',' PROTOCOLE NET**
(29)	**| AUTORISER ACCEE ':' adripv LAN '-' '>' adripv ',' CONSTANT ',' PROTOCOLE NET**
(34)	**| AUTORISER ACCEE ':' LAN '-' '>' ZONE CONSTANT**
(35)	**| AUTORISER ACCEE ':' LAN '-' '>' adripv ZONE CONSTANT**
(37)	**| AUTORISER ACCEE ':' adripv LAN '-' '>' ZONE CONSTANT**
(38)	**| AUTORISER ACCEE ':' adripv LAN '-' '>' adripv ZONE CONSTANT**
(43)	**| PUBLIER IDFS ZONE CONSTANT**
(44)	**| PUBLIER IDFS ZONE CONSTANT ':' CONSTANT**

L'ensemble de ces règles, permet soit :

- de publier un service au Net ;
- d'autoriser un accès d'une zone à l'autre.

Note : Nous n'avons pas cité toutes les règles de configuration.

2.6. Conclusion :

Dans ce chapitre, nous avons fait un petit tour sur la description des règles grammaticales et syntaxiques du langage proposé qui désigne la première étape pour concevoir un tel préprocesseur.

Partie III

Implémentation

Chapitre 1

Programmation du préprocesseur

Dans ce chapitre :

- Introduction
- Description des fichiers LEX et YACC
  - ✓ Les différentes parties du fichier LEX
  - ✓ Les différentes parties du fichier Yacc
- Schéma de compilation
- Conclusion

1.1. Introduction :

Le processus de programmation qu'on a suivi passe par trois étapes successives, la première étape est la construction des fichiers sources Lex et Yacc. La deuxième étape nous amène à obtenir les deux fichiers objets de l'analyseur lexicale et syntaxique après l'utilisation des compilateurs Lex, Yacc et GCC, la dernière étape sert à faire l'édition de liens entre les différents fichiers objets pour obtenir le fichier exécutable (le préprocesseur).

Dans ce chapitre on va parler sur le détaille des fichiers Lex et Yacc, et leurs schéma de compilation.

1.2. Description des fichiers Lex et Yacc :

1.2.1. Les différentes parties du fichier Lex :

Le fichier *al.l* contient la description du source **Lex** de l'analyseur lexical, il est formé des deux premières parties d'un fichier Lex (la partie **déclarations** suivi de la partie **règles**).

```
%{
#include "y.tab.h"
#include "global.h"
int li=1,mot=0;
%}
```

C'est la partie déclarations écrite dans la syntaxe du langage 'c', elle englobe la déclaration des fichiers entêtes nécessaires a l'étape de l'édition de liens, suivi de la déclaration des variables globales au fichier Lex utilisées dans le reste du source.

```
chiffre [0-9]
consta [0-9]|[1-9][0-9]|[1-2][0-4][0-9]|(1)[5-9][0-9]|(25)[0-5]
adrip {consta}\.{consta}\.{consta}\.{consta}
plageadrip {adrip}\:{adrip}
constm [0-9]|[1-2][0-9]|(3)[0-2]
adrr  {adrip}\/{constm}
constant ({chiffre}){1,5}
lettre [A-Za-z]
protocole  "tcp"|"udp"
idfiface "eth"({constant})
idfs {lettre}({lettre}|{chiffre})*
```

C'est la suite de la partie déclarations contenant la définition des expressions régulières écrite dans la syntaxe du compilateur Lex.

```
%%
"DECLARATION"  { mot++; return(DECLARATION); }
[\n] { strcpy(yylval.chaine,"ret");  li++;  mot=0; }
[ \t] ;
"CONFIGURATION" { fprintf(yyout,"#!/bin/sh \n"); fprintf(yyout,"\n");
fprintf(yyout,"iptables -X\n");//supression des chaines utilisateur
fprintf(yyout,"iptables -F FORWARD\n");//vider la chaine forward
...
fprintf(yyout,"iptables -t mongle -P OUTPUT DROP\n");
fprintf(yyout,"iptables -F PREROUTING -t mongle\n");
fprintf(yyout,"iptables -t mongle -P PREROUTING DROP\n");
mot++;return(CONFIGURATION);}
"FW" {  mot++;  return(FW); }
"ZONE" { strcpy(yylval.chaine,yytext);  mot++;  return(ZONE);  }
"NET"  { mot++; return(NET);   }
"ZONES"  {  mot++;  return(ZONES);   }
"SERVICES"  {    mot++;    return(SERVICES);    }
"AUTORISER"  {   mot++; return(AUTORISER); }
"ACCEE"  {  mot++; return(ACCEE);   }
"PUBLIER"  {   mot++;  return(PUBLIER);  }
"LAN" { strcpy(yylval.chaine,yytext);
...
{idfiface} { strcpy(yylval.chaine,yytext); mot++;      return(IDFIFACE);  }
{idfs} { strcpy(yylval.chaine,yytext);   mot++; return(IDFS);  }
[\{\}\:\/\(\)\-\>\,] {   mot++; return(yytext[0]); }
.   { mot++;     yyerror();   }
```

C'est la partie règles de traduction, dans la quelle des actions écrites en langage 'c' sont associées pour chaque expression régulière.

1.2.2 Les différentes parties du fichier Yacc :

```
%{
#include "global.h"
#define max1 99998 /*nbre max de zones*/
#define max2 99998 /*nbre max de services-1*/
int i=0, j=0,n=-1;
/* la declaration d'une structure contenant les informations necessaires
pour la description d'une zone */
struct descriptzone {
char idfzone[4];/* une DMZ est caracterisee par un n°*/
```

```
char ar[18];   /* sauf pour le cas : LAN */
char aip[16];
char idfiface[7];
};
/* la declaration d'une structure contenant les informations
necessaires pour la description d'un service*/
struct descriptserv {
            char idfserv[21];/* exple:WEBZONE1:au max 13 car*/
          char aip[16];   /*pour idf d'un service purement*/
          char constp[6];
          char prot[4];
          };
/* la declaration d'une structure contenant les informations
necessaires pour la description de l'interface publique*/
struct descriptiface {
            char aip[16];
            char idf[7];/*"eth" siuvi de trois chiffres au max*/
            };
 struct descriptzone tab1[max1];/*table de symboles des zones : LAN et
DMZs*/
struct descriptserv tab2[max2];/*table de symboles pours les services*/
struct descriptiface ifacenet;/*variable globale contenant la description*/
                /*de l'interface publique*/
%}
```

La partie déclarations du fichier Yacc contenant la déclaration des fichiers entêtes nécessaires à l'étape d'édition de lien suivi de la déclaration des variables globales au fichier Yacc utilisées dans le reste du source.

```
%union {  char chaine[21];    } ;
%token <chaine> DECLARATION CONFIGURATION FW ZONE LAN ADRIP
%token <chaine> IDFIFACE CONSTANT NET ZONES ADRR SERVICES IDFS
                CONSTP
%token <chaine> AUTORISER ACCEE PROTOCOLE PLAGEADRIP PUBLIER
%%
```

C'est la déclaration du type de données du terminal courant par le mot clé **%union**, suivi de la déclaration des terminaux par le mot clé **%token.**

```
lang : DECLARATION declar CONFIGURATION conf
       { fprintf(yyout,"iptables-save \n" );
         fprintf(yyout,"iptables -L \n" );
         fprintf(yyout,"iptables -t nat-L \n" );
         fprintf(yyout,"\n" ) ; } ;
declar : fw zone service   ;
fw : FW '{' interfacez interfacenet '}'  ;
interfacez : interfacez1 interfacez2   ;
interfacez1 : LAN ':' ADRIP ',' IDFIFACE
          { strcpy( tab1[i].idfzone,$1);
            strcpy(tab1[i].aip,$3);
            strcpy(tab1[i].idfiface,$5);
            i++;}  ;
interfacez2 : ZONE CONSTANT':' ADRIP ',' IDFIFACE ifaceplus
         { strcpy(tab1[i].idfzone,$2);
           strcpy(tab1[i].aip,$4);
           strcpy(tab1[i].idfiface,$6);
           i++; } ;
ifaceplus : /* rein */ |interfacez2     ;
interfacenet : NET ':' ADRIP ',' IDFIFACE
         { strcpy(ifacenet.aip,$3);
           strcpy(ifacenet.idf,$5);  }  ;
zone : ZONES '{' zone1 zone2 '}'  ;
zone1 :LAN ':' ADRR
        { n=chercher1(tab1 ,max1,$1);
          strcpy(tab1[n].ar,$3);  n=-1; } ;
zone2 : ZONE CONSTANT ':' ADRR suiteplus
        { n=chercher1(tab1,max1,$2);
          strcpy(tab1[n].ar,$4);  n=-1;  }  ;
suiteplus : /* rien */  | zone2    ;
service : SERVICES '{' servicez '}'   ;
servicez : IDFS ZONE CONSTANT ':' ADRIP ',' CONSTANT ',' PROTOCOLE
             serviceplus
      { strcat($1,$2);
        strcat($1,$3);
        strcpy(tab2[j].idfserv,$1);
        strcpy(tab2[j].aip,$5);
        strcpy(tab2[j].constp,$7);
        strcpy(tab2[j].prot,$9);
        j++; }  ;
serviceplus : /* rien */  | servicez     ;
conf : confzone confplus   ;
confplus :/* rien */ |conf   ;
```

Dans cette partie les différents règles grammaticales de la section de déclaration du langage source proposée ainsi que les actions syntaxiques et sémantiques correspondantes sont cités.

La section en dessous contient la description de quelque règles grammaticales de ainsi que les actions syntaxiques et sémantiques correspondantes concernant la section de configuration du langage source proposée.

❖ **Autoriser l'accès du LAN au Net :**

```
confzone : AUTORISER ACCEE ':' ADRIP LAN '-' '>' CONSTANT ',' PROTOCOLE
           NET
```

Cette règle permet d'autoriser un poste du réseau local caractérisé par son adresse IP à naviguer dans le Net sur un port bien précis, et pour un type de connexion bien définie.

```
{ int nlan=chercher1(tab1,max1,$5);
fprintf(yyout,"iptables -t nat -A POSTROUTING -o %s -s %s -p %s -j SNAT --to-
source %s:%s \n",ifacenet.idf,$4,$10,ifacenet.aip,$8 );

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -m state --state
NEW,ESTABLISHED -j ACCEPT \n",tab1[nlan].idfiface,ifacenet.idf,$4);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -m state --state
ESTABLISHED -j ACCEPT\n",ifacenet.idf,tab1[nlan].idfiface,$4);

fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

```
| AUTORISER ACCEE ':' ADRIP LAN '-'">' ADRIP ',' CONSTANT ','PROTOCOLE
NET
```

Cette règle permet d'autoriser un poste du réseau local caractérisé par son adresse IP à naviguer dans un poste du Net caractérisé par son adresse, sur un port bien précis, pour un type de connexion bien définie.

```
{ int nlan=chercher1(tab1,max1,$5);
fprintf(yyout,"iptables -t nat -A POSTROUTING -o %s -s %s -d %s -p %s -j SNAT –to
-source %s:%s \n",ifacenet.idf,$4,$8,$12,ifacenet.aip,$10);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -d %s -m state --state
NEW,ESTABLISHED -j ACCEPT \n",tab1[nlan].idfiface,ifacenet.idf,$4,$8);

 fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -m state --state
ESTABLISHED -j ACCEPT \n",ifacenet.idf,tab1[nlan].idfiface,$8,$4);

fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

| AUTORISER ACCEE ':' LAN '-' '>' CONSTANT ',' PROTOCOLE NET

Cette règle permet d'autoriser l'ensemble des postes du réseau local à naviguer dans le Net sur un port bien précis, et pour un type de connexion bien définie.

```
{ int nlan=chercher1(tab1,max1,$4);
fprintf(yyout,"iptables -t nat -A POSTROUTING -o %s -s %s -p %s -j SNAT --to-
source %s:%s\n",ifacenet.idf,tab1[nlan].ar,$9,ifacenet.aip,$7);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -m state --state
NEW,ESTABLISHED -j ACCEPT \n",tab1[nlan].idfiface, ifacenet.idf,
tab1[nlan].ar);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -m state --state
ESTABLISHED -j ACCEPT \n",ifacenet.idf,tab1[nlan].idfiface,tab1[nlan].ar);

fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

| AUTORISER ACCEE ' :' LAN '-">' ADRIP ',' CONSTANT ',' PROTOCOLE NET

Cette règle permet d'autoriser l'ensemble des postes du réseau local à naviguer dans un poste du Net caractérisé par son adresse IP et sur un port bien précis, pour un type de connexion bien définie.

```
{ int nlan=chercher1(tab1,max1,$4);
fprintf(yyout,"iptables -t nat -A POSTROUTING -o %s -s %s -d %s -p %s -jSNAT --
to-source %s:%s \n",ifacenet.idf,tab1[nlan].ar,$7,$11,ifacenet.aip,$9);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -d %s -m state --state

NEW,ESTABLISHED -j ACCEPT\n",
tab1[nlan].idfiface,ifacenet.idf,tab1[nlan].ar,$7);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -m state --state
ESTABLISHED -j ACCEPT \n",ifacenet.idf,tab1[nlan].idfiface,$7,tab1[nlan].ar);
fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

| AUTORISER ACCEE ':' LAN '-">' PROTOCOLE NET

Cette règle permet d'autoriser l'ensemble des postes du réseau local à naviguer dans tous les postes du Net, sans aucune restriction sur le numéro de port, et pour un type de connexion bien définie.

```
{ int nlan=chercher1(tab1,max1,$4);
fprintf(yyout,"iptables -t nat -A POSTROUTING -o %s -s %s -p %s -j SNAT --to-
source %s\n",ifacenet.idf,tab1[nlan].ar,$7,ifacenet.aip);

fprintf(yyout,"iptables -A FORWARD -i %s-o %s -s %s -m state --state
NEW,ESTABLISHED -j ACCEPT\n",tab1[nlan].idfiface, ifacenet.idf,
tab1[nlan].ar);

    fprintf(yyout,"iptables -A FORWARD -i %s -o %s-d %s -m state --state
ESTABLISHED -j ACCEPT\n ",ifacenet.idf,tab1[nlan].idfiface,tab1[nlan].ar);
fprintf(yyout," \n" );   }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

❖ **Autoriser l'accès du LAN au DMZ :**

| AUTORISER ACCEE ':' ADRIP LAN '-">'ADRIP ZONE CONSTANT

Cette règle permet d'autoriser l'accès d'un poste du LAN caractérisé par son adresse IP d'accéder à un poste de la DMZ caractérisée par son adresse IP.

```
{ int nlan=chercher1(tab1,max1,$5);
int nzone=chercher1(tab1,max1,$10);

 fprintf(yyout,"iptables  -A FORWARD -i %s -o %s -s %s -d %s  -m state --state
NEW,ESTABLISHED -j ACCEPT\n", tab1[nlan].idfiface, tab1[nzone].idfiface, $4,
$8);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -d %s -m state --state
ESTABLISHED -j ACCEPT\n",tab1[nzone].idfiface,tab1[nlan].idfiface,$8,$4);

fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

```
| AUTORISER ACCEE ':' LAN '-' '>' ZONE CONSTANT
```

Cette règle permet d'autorisée l'accès de l'ensemble des postes du réseau local à tout l'ensemble de postes de la DMZ sans aucune restriction.

```
{int nlan=chercher1(tab1,max1,$4);
int nzone=chercher1(tab1,max1,$8);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -d %s -m state --state
NEW,ESTABLISHED-j ACCEPT\n", tab1[nlan].idfiface, tab1[nzone].idfiface,
tab1[nlan].ar, tab1[nzone].ar);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -d %s -m state --state
ESTABLISHED -j ACCEPT \n",tab1[nzone].idfiface, tab1[nlan].idfiface,
tab1[nzone].ar, tab1[nlan].ar);

fprintf(yyout," \n" ); }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

❖ **Publier les services de la DMZ :**

```
| PUBLIER IDFS ZONE CONSTANT
```

Cette règle permet de publier un service caractérisé par son identificateur et la zone démilitarisée qui l'héberge.

```
{ int nzone=chercher1(tab1,max1,$4);
strcat($2,$3);
strcat($2,$4);
int nserv=chercher2(tab2,max2,$2);

fprintf(yyout,"iptables -t nat -A PREROUTING -i %s -d %s -p %s --dport %s -j
DNAT--to-destination %s \n",ifacenet.idf, ifacenet.aip,tab2[nserv].prot,
tab2[nserv].constp, tab2[nserv].aip);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -p %s --dport %s -m state --
state NEW,ESTABLISHED -j ACCEPT \n",ifacenet.idf, tab1[nzone].idfiface,
tab2[nserv].aip, tab2[nserv].prot, tab2[nserv].constp);
```

```
fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -p %s --sport %s -m state --
state ESTABLISHED -j ACCEPT \n",tab1[nzone].idfiface,
ifacenet.idf,tab2[nserv].aip, tab2[nserv].prot,tab2[nserv].constp );

fprintf(yyout," \n" );    }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

| PUBLIER IDFS ZONE CONSTANT ':' CONSTANT

Cette règle permet de publier un service caractérisé par son identificateur et la zone démilitarisée qui l'héberge sur un numéro de port bien précis.

```
{ int nzone=chercher1(tab1,max1,$4);
strcat($2,$3);
strcat($2,$4);
int nserv=chercher2(tab2,max2,$2);

fprintf(yyout,"iptables -t nat -A PREROUTING -i %s -d %s -p %s --dport %s -j
DNAT --to-destination %s:%s \n",ifacenet.idf,ifacenet.aip, tab2nserv].prot,$6,
tab2[nserv].aip, tab2[nserv].constp);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -d %s -p %s --dport %s -m state --
state NEW,ESTABLISHED -j ACCEPT \n", ifacenet.idf, tab1[nzone].idfiface,
tab2[nserv].aip, tab2[nserv].prot,tab2[nserv].constp);

fprintf(yyout,"iptables -A FORWARD -i %s -o %s -s %s -p %s --sport %s -m state --
state ESTABLISHED -j ACCEPT \n",tab1[nzone].idfiface, ifacenet.idf,
tab2[nserv].aip, tab2[nserv].prot,tab2[nserv].constp );

fprintf(yyout," \n" );    }
```

Ce bloc d'instructions permet d'afficher sur le fichier de sortie l'ensemble des règles IPtables correspondantes à la règle écrite en dessus.

```
int chercher1( struct descriptzone tab[],int n,char name[])
  {
    int count=0;
    for(;count<n;count++)
          {
            if((strcmp(name,tab[count].idfzone))==0) break;
          }
    if (count==max1)
          {
              fprintf(stderr,"\n");
```

```
                fprintf(stderr,"Erreur : des informations liées a la zone %s
                                    manquent.\n",name);
                fprintf(stderr,"\n");
                exit (1);
            }
    }
```

La fonction *cherche1 ()* permet de chercher les coordonnées d'une zone (LAN et DMZ) dans la table de description des zones.

```
int chercher2( struct descriptserv tab[],int n,char name[])
{
    int count=0;
    for(;count<n;count++)
        {
        if(strcmp( name,tab[count].idfserv)==0) break;
        }
if(count==max2)
        {
           fprintf(stderr,"\n");

           fprintf(stderr,"Erreur : le service ( %s ) n'est pas déclarer.\n",name);

           fprintf(stderr,"\n");
           exit (1);
        }
}
```

La fonction *cherche2 ()* permet de chercher les coordonnées d'un service dans la table de description des services.

```
int yyerror(char *s)

  {
     if((strcmp(yylval.chaine,"ret")==0)&&(li==2)) {li--;}
     fprintf(stderr,"\n");

     fprintf(stderr,"Erreur dans la ligne numéro %d et le mot numéro %d : ( %s )
                        \n",li,mot,yytext);

     fprintf(stderr,"\n");
     return 0;

  }
```

La fonction *yyerror()* permet de localiser l'erreur dans le fichier d'entrée saisie par l'utilisateur.

```
int main(int argc,char *argv[])
{
    if(argc<=1)
    fprintf(stderr,"Argument manquant.\n");

    yyin=fopen(argv[1],"r");

    if(yyin==NULL)
    fprintf(stderr,"Erreur d'ouverture du fichier d'entree.\n");

    yyout=fopen(argv2],"w");

    if(yyout==NULL)
    fprintf(stderr,"Erreur d'ouverture du fichier de sortie.\n");

   yyparse();
   fclose (yyin);
   fclose(yyout);
   return 0;
}
```

La fonction ***main()*** lance l'analyseur syntaxique *pan* un appelle à la fonction ***yyparse ()***, et décide de l'ouverture et la fermeture des fichiers d'entrées et de sortie,etc.

Nous avons définie deux autres fichiers nommés global.h et ***Makefile*** :

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>


extern int li;

extern int mot;

extern int yyleng;

extern char *yytext;

extern FILE *yyin;

extern FILE *yyout;
```

Le fichier ***global.h*** contient la déclaration des fichiers entêtes et des variables globales utilisées dans les deux fichiers sources Lex et Yacc.

```
CC = gcc

CFLAGS = -Wall

LDFLAGS = -ll

all : yacc lex compiler

lex : lex.yy.c y.tab.h

yacc : y.tab.c

lex.yy.c : al.l

        lex al.l

y.tab.c : as.y

yacc -d as.y

y.tab.o :  y.tab.c

       ${CC} –c y.tab.c

lex.yy.o : lex.yy.c  y.tab.h

       ${CC} -c  lex.yy.c

Compiler : lex.yy.o  y.tab.o

       ${CC} -o compiler lex.yy.o  y.tab.o

       ${LDFLAGS}
```

Le fichier ***Makefile*** contient la suite de commandes de compilation et d'édition de liens nécessaires pour obtenir l'éxécuable du préprocesseur par la simple commande ***make***.

1.3. Schéma de compilation

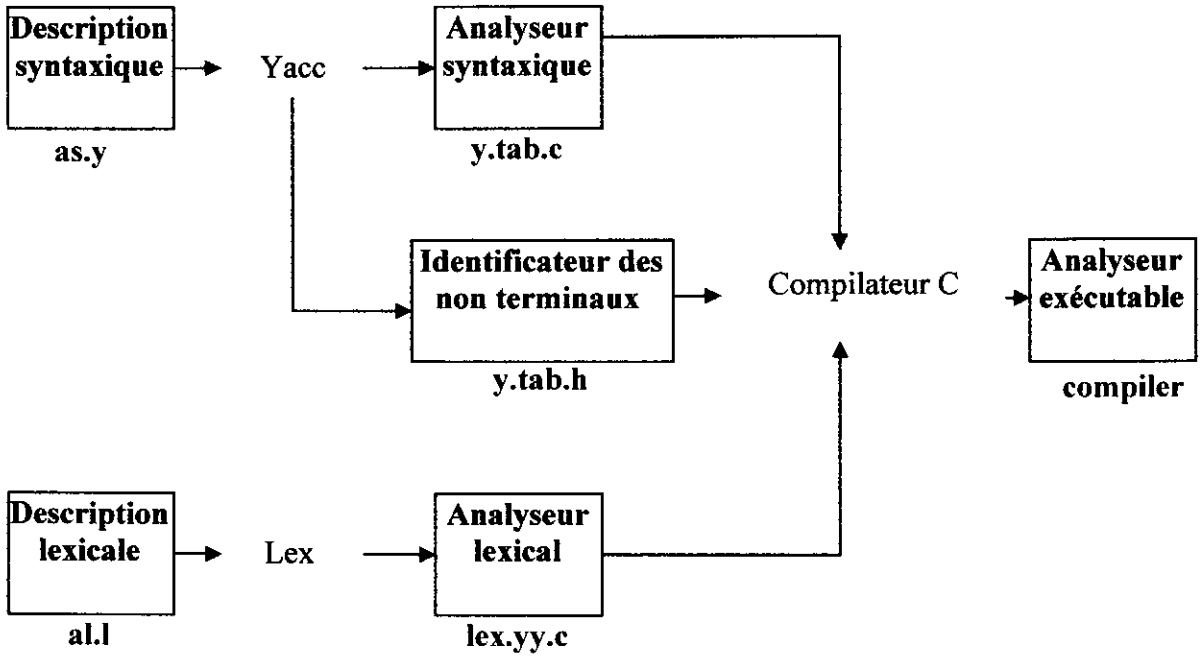


Figure III.1.1 : le schéma de compilation du source Lex et Yacc.

Pour arriver à obtenir l'exécutable du préprocesseur il faut suivre les étapes suivantes :

- Construire les fichiers nommées : ***al.l***, ***as.y*** contenant la description des analyseurs lexical et syntaxique, et global.h contenant la description des variables globales ;
- Utiliser le compilateur Yacc pou obtenir le code 'c' de l'analyseur syntaxique (***y.tab.c***), et le ficher entête ***y.tab.h*** ;
- Utiliser le compilateur Lex pou obtenir le code 'c' de l'analyseur lexical (***lex.yy.c***) ;
- Utiliser le compilateur 'c' pour obtenir le fichier **compiler** à partir des trois fichiers **lex.yy.c**, **y.tab.c** et **y.tab.h**.

Le fichier Makefile permet de générer l'ensemble des commandes suivantes :

```
> yacc -d as.y
> lex al.l
> gcc y.tab.c lex.yy.c -ll -o Compiler
```

1.3. Conclusion:

Dans ce chapitre, on a vue les différentes parties des fichiers Lex et Yacc ainsi que l'interaction entre eux. Le fichier Lex contient la description de l'analyseur lexical et le fichier Yacc contient la description de l'analyseur syntaxique. L'analyse sémantique est garantie par les actions sémantiques dans les règles grammaticales du fichier Yacc.

Chapitre 2

Test du préprocesseur

Dans ce chapitre :

- Introduction
- Tests et résultats
- Conclusion

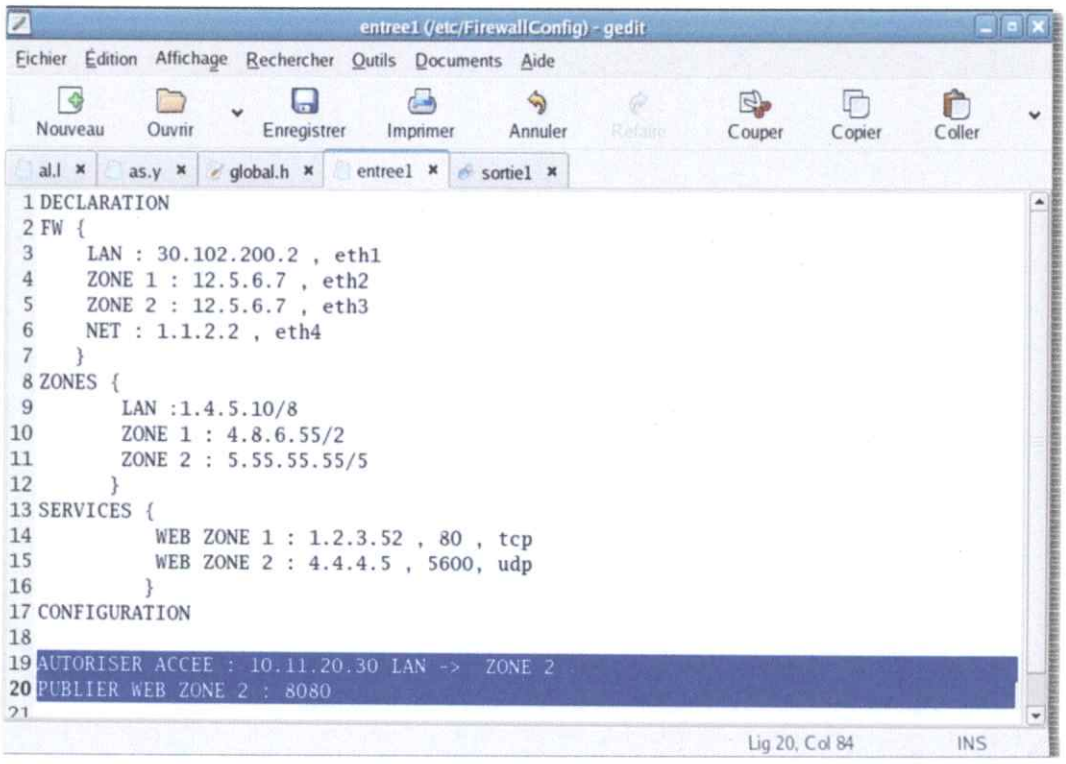
1.1. Introduction :

Notre travail consistait à réaliser un préprocesseur qui à pour objectif la génération automatique des commandes IPtables correspondantes a la configuration du Firewall pour l'implémentation de l'architecture de sécurité en DMZ. Sur la base de cet objectif, nous avons effectué les tests nécessaires pour montrer son fonctionnement.

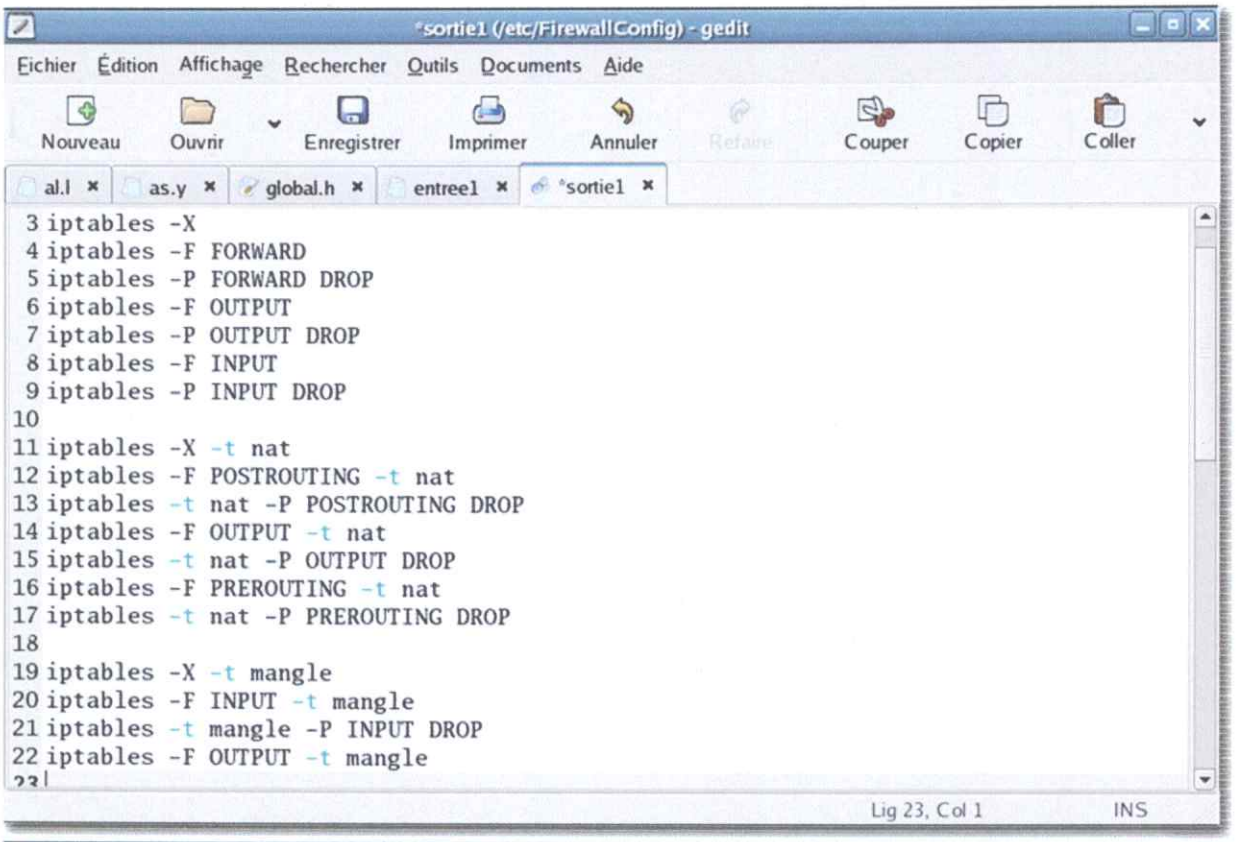
1.2. Tests et résultats :

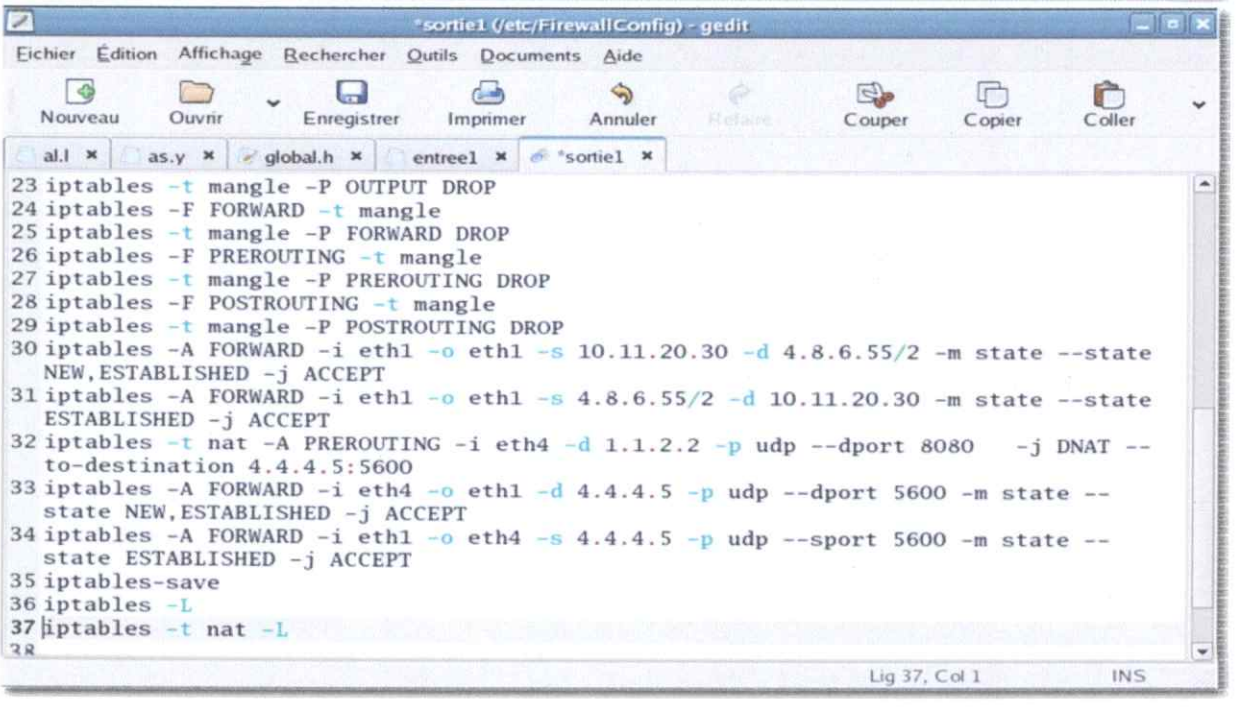
- La première partie du test à pour objectif de montrer l'utilité de ce logiciel partant des points de vues suivants :

  - Un langage « parlant » justifié par l'utilisation de mots clé significatifs tel que « LAN », « AUTORISER », « PUBLIER » etc., et des flèches indiquant le sens d'accès.
  - Une minimisation importante du code source de la configuration par rapport à une configuration assurée par les commandes IPtables.



Les deux règles sélectionnées dans la prise d'écran en dessous qui représentent le fichier d'entrée, génèrent l'ensemble des commandes IPtables apparue dans les deux prises d'écran suivantes :

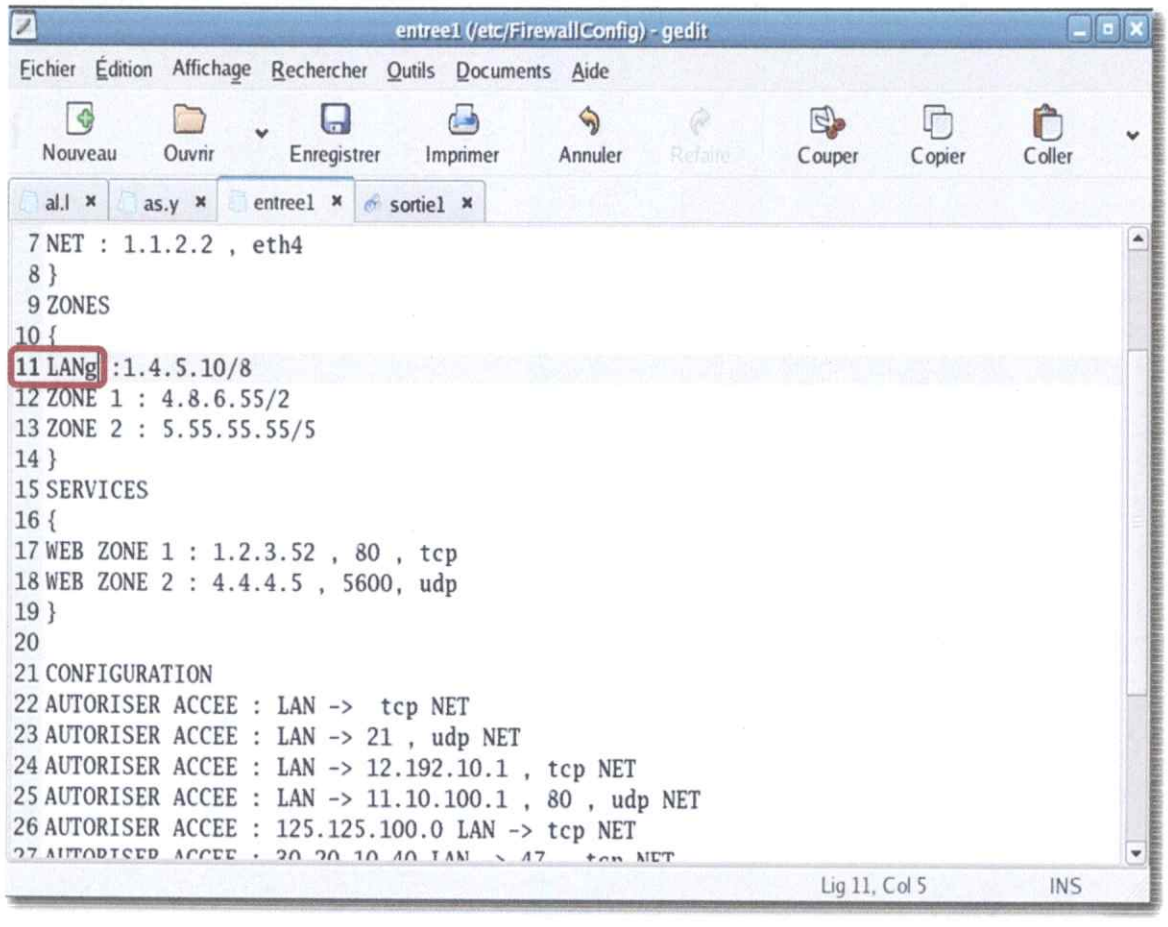




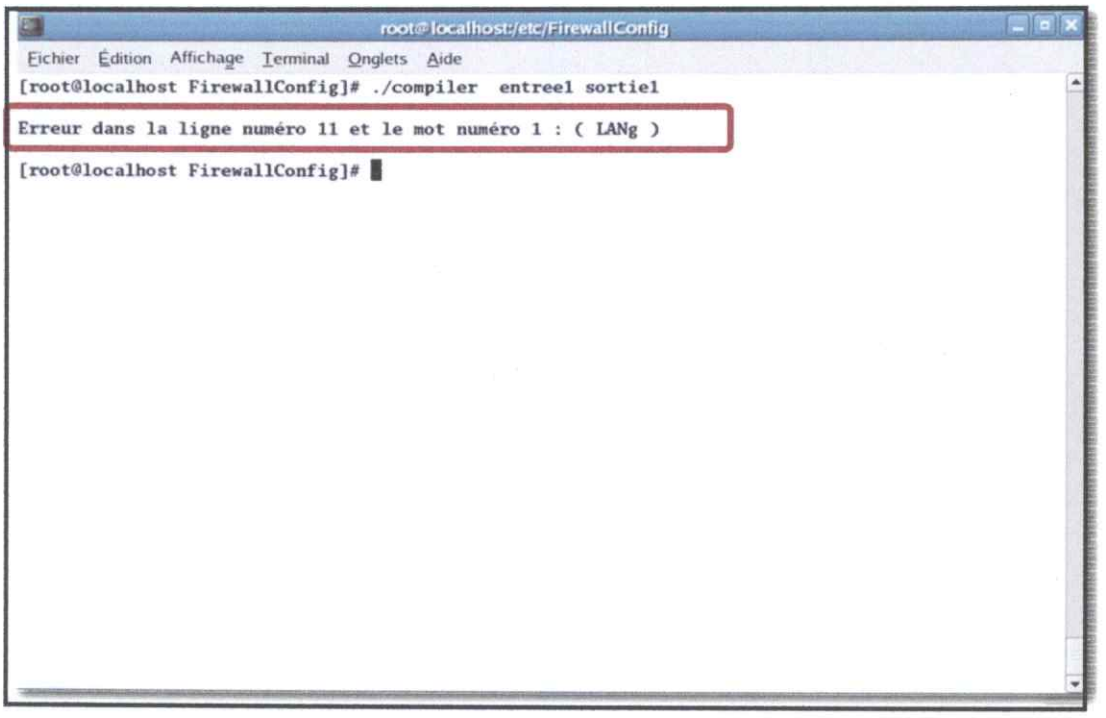
On peut observer que la taille du fichier source écrit dans notre langage est un peu la plus grande que la taille du fichier des règles IPtables, mais dès que les règles d'autorisation et/ou publication augmentent, la taille du fichier des règles IPtables générées devient plus important.

- La deuxième partie du test à pour objectif, la présentation de quelques exemples de détection des erreurs dans le code source saisie :

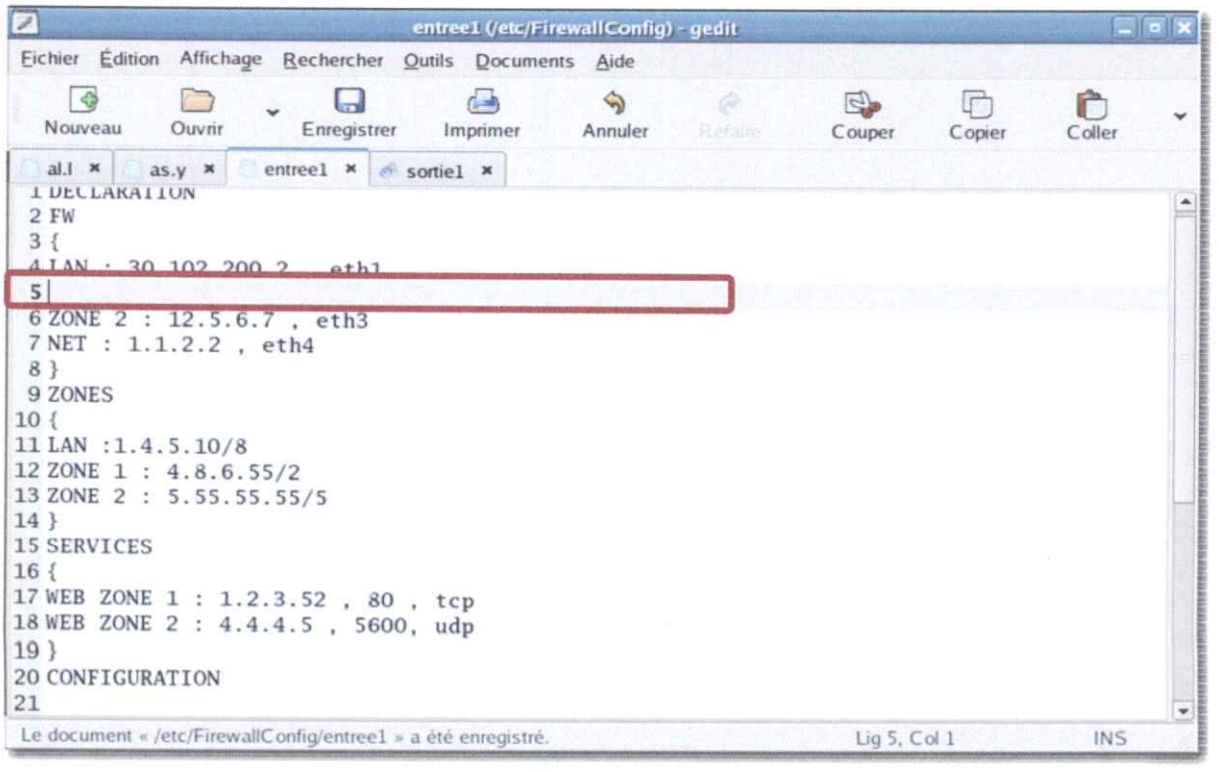
  - La prise d'écran suivante, représente le ficher d'entrée, elle contient une erreur dans le mot clé « LAN » :



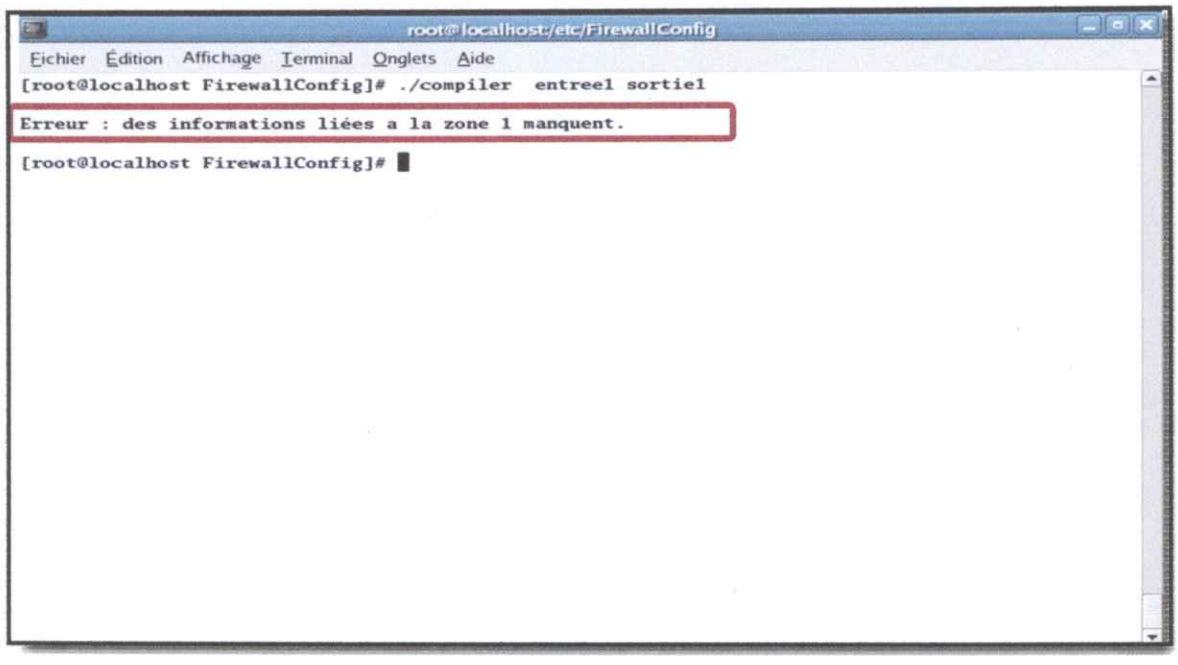
Le préprocesseur affiche un message d'erreur avec le numéro de ligne et sa position, ainsi que le mot lui-même.



- La prise d'écran suivante, représente un texte du ficher d'entrée avec une erreur de non déclaration des attributs de « ZONE 1 » :



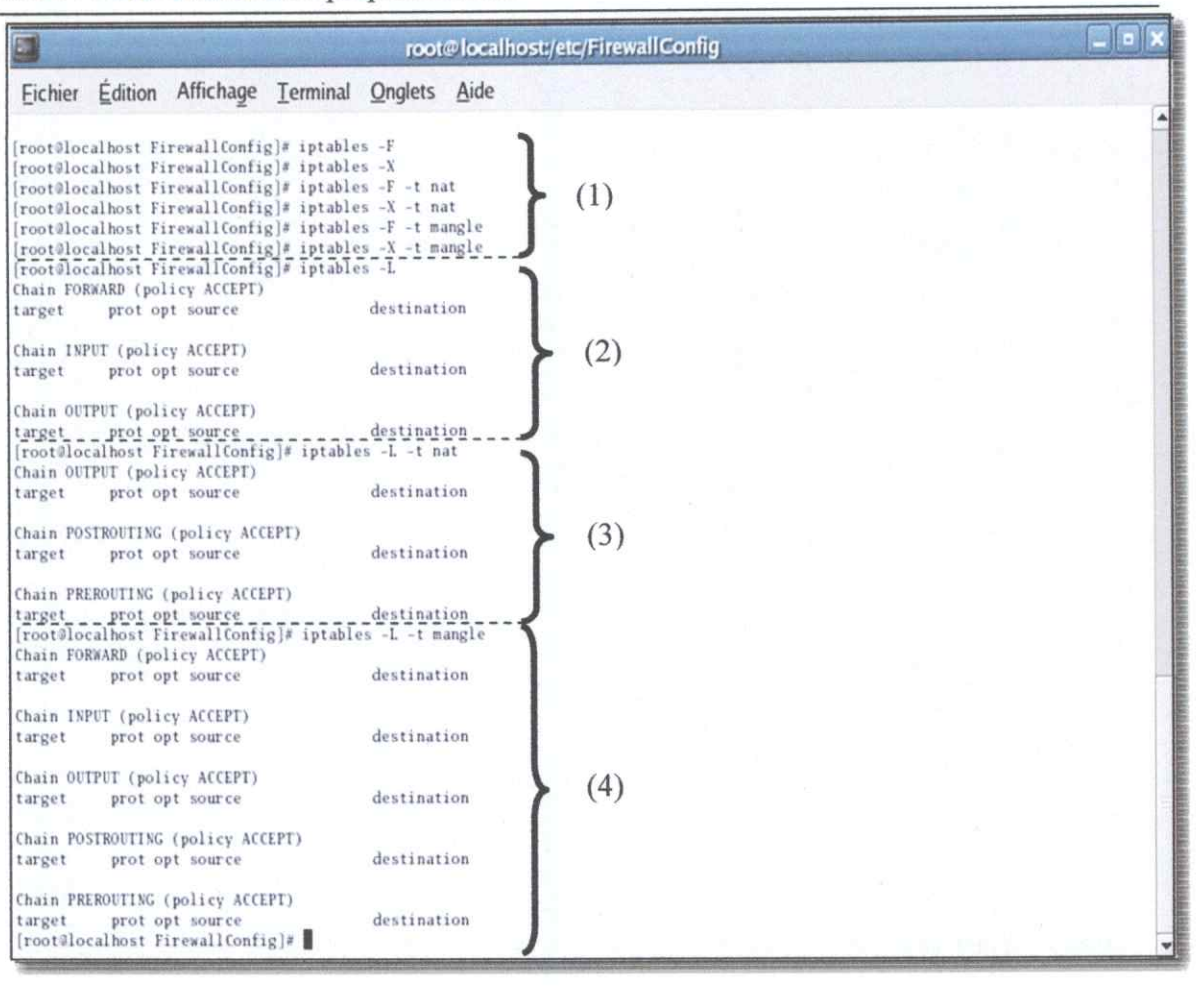
L'utilisation d'une zone ou bien d'un service non déclaré, permet au préprocesseur d'afficher un message d'erreur indiquant un manque d'informations concernant cette zone ou ce service :



- La troisième partie du test à pour objectif de montrer la validité des règles IPtables générées automatiquement par le préprocesseur au niveau du noyau de Linux :

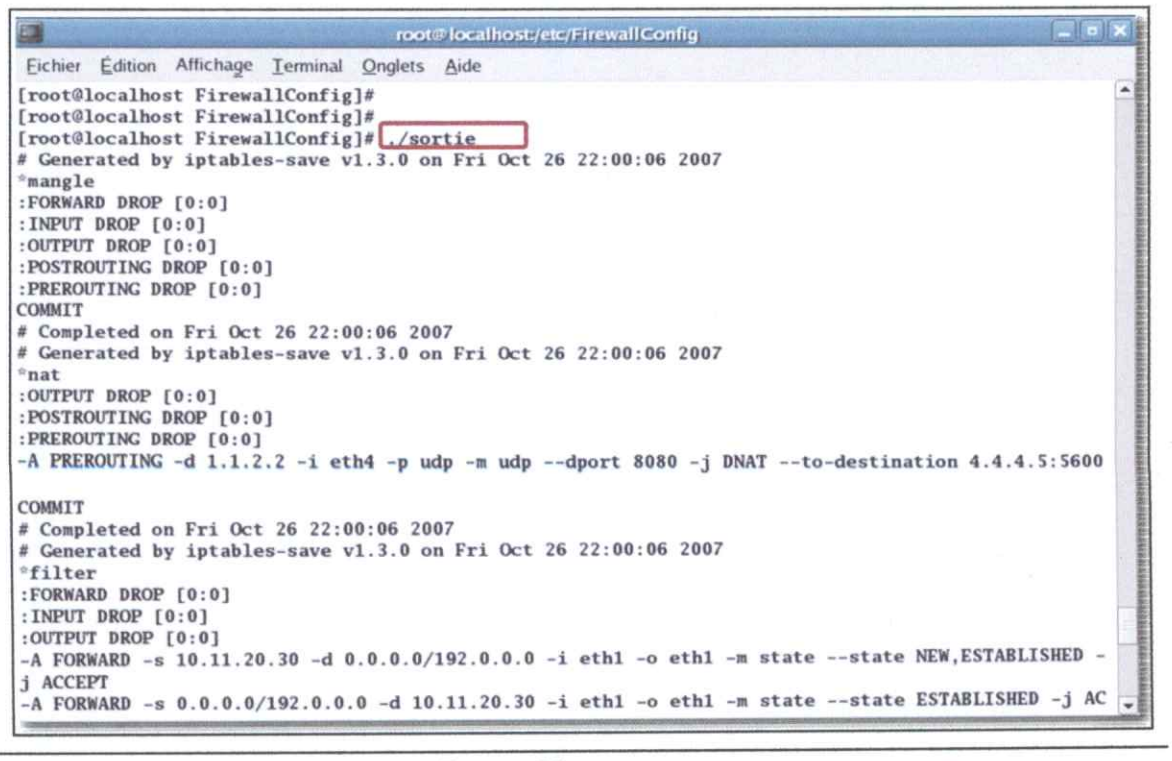
  - Le premier bloque de la prise d'écran suivante contient les règles IPtables nécessaires pour vider les trois tables **Filter, Nat** et **Mangle :**

    - Bloque numéro (2) : montrer que la tables **Filter** est vide.
    - Bloque numéro (3) : montrer que la tables **Nat** est vide.
    - Bloque numéro (4) : montrer que la tables **Mangle** est vide.

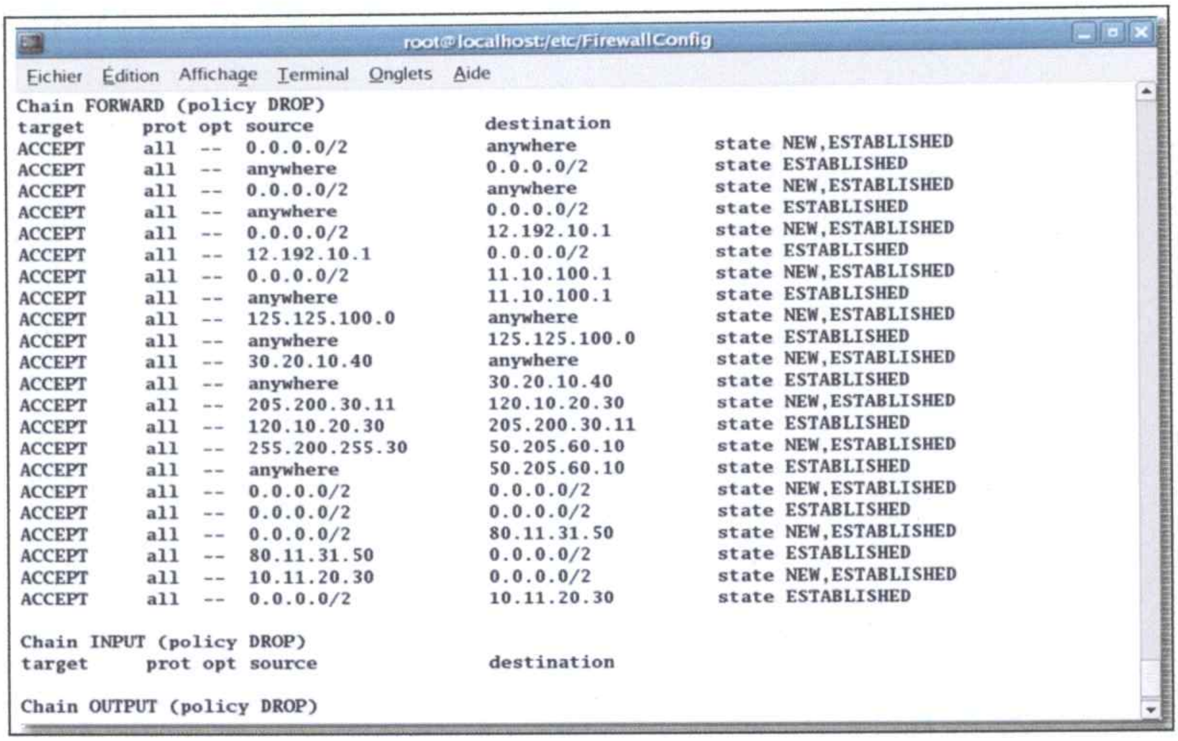


Après avoir vider les trois tables, on applique la commande « ./sortie » qui permet de valider au niveau du noyau Linux, les règles IPtables écrites dans le fichier « **sortie** »

.



La prise d'écran suivante montre que la table **Filter** a été remplie :

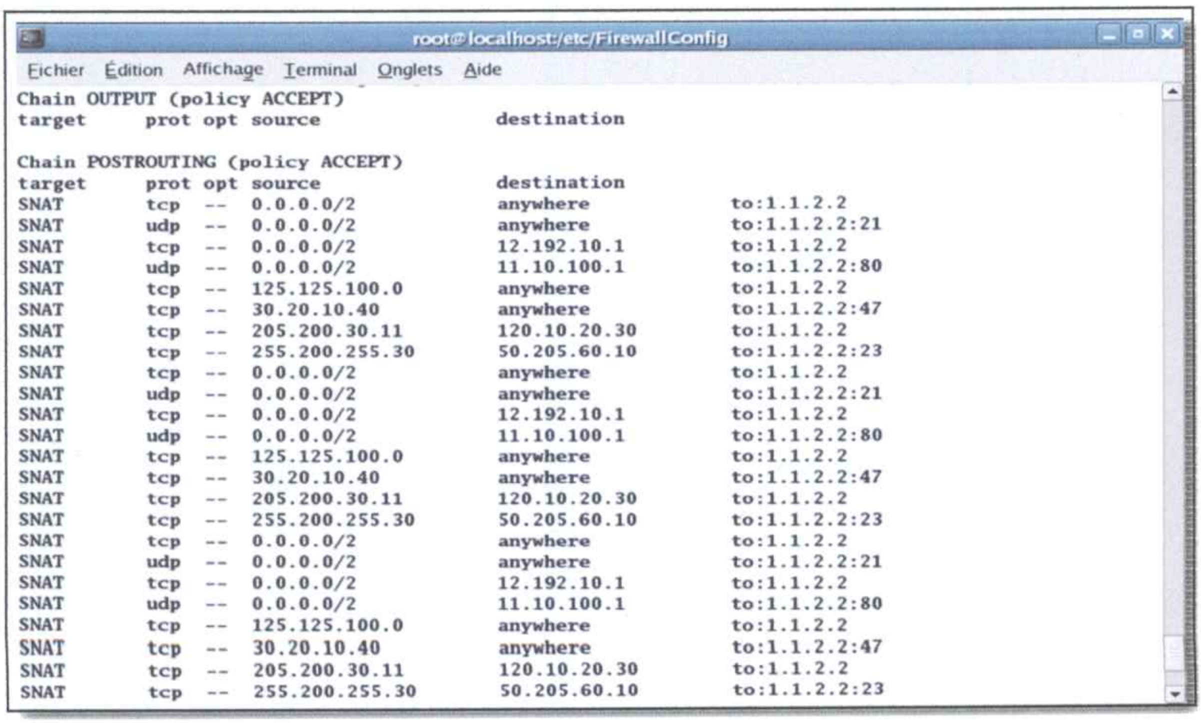


```
root@localhost:/etc/FirewallConfig
Fichier  Édition  Affichage  Terminal  Onglets  Aide
Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/2            anywhere             state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             0.0.0.0/2            state ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            anywhere             state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             0.0.0.0/2            state ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            12.192.10.1          state NEW,ESTABLISHED
ACCEPT     all  --  12.192.10.1          0.0.0.0/2            state ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            11.10.100.1          state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             11.10.100.1          state ESTABLISHED
ACCEPT     all  --  125.125.100.0        anywhere             state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             125.125.100.0        state ESTABLISHED
ACCEPT     all  --  30.20.10.40          anywhere             state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             30.20.10.40          state ESTABLISHED
ACCEPT     all  --  205.200.30.11        120.10.20.30         state NEW,ESTABLISHED
ACCEPT     all  --  120.10.20.30         205.200.30.11        state ESTABLISHED
ACCEPT     all  --  255.200.255.30       50.205.60.10         state NEW,ESTABLISHED
ACCEPT     all  --  anywhere             50.205.60.10         state ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            0.0.0.0/2            state NEW,ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            0.0.0.0/2            state ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            80.11.31.50          state NEW,ESTABLISHED
ACCEPT     all  --  80.11.31.50          0.0.0.0/2            state ESTABLISHED
ACCEPT     all  --  10.11.20.30          0.0.0.0/2            state NEW,ESTABLISHED
ACCEPT     all  --  0.0.0.0/2            10.11.20.30          state ESTABLISHED

Chain INPUT (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
```

La prise d'écran suivante montre que la table **Nat** a été remplie :



```
root@localhost:/etc/FirewallConfig
Fichier  Édition  Affichage  Terminal  Onglets  Aide
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       tcp  --  0.0.0.0/2            anywhere             to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            anywhere             to:1.1.2.2:21
SNAT       tcp  --  0.0.0.0/2            12.192.10.1          to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            11.10.100.1          to:1.1.2.2:80
SNAT       tcp  --  125.125.100.0        anywhere             to:1.1.2.2
SNAT       tcp  --  30.20.10.40          anywhere             to:1.1.2.2:47
SNAT       tcp  --  205.200.30.11        120.10.20.30         to:1.1.2.2
SNAT       tcp  --  255.200.255.30       50.205.60.10         to:1.1.2.2:23
SNAT       tcp  --  0.0.0.0/2            anywhere             to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            anywhere             to:1.1.2.2:21
SNAT       tcp  --  0.0.0.0/2            12.192.10.1          to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            11.10.100.1          to:1.1.2.2:80
SNAT       tcp  --  125.125.100.0        anywhere             to:1.1.2.2
SNAT       tcp  --  30.20.10.40          anywhere             to:1.1.2.2:47
SNAT       tcp  --  205.200.30.11        120.10.20.30         to:1.1.2.2
SNAT       tcp  --  255.200.255.30       50.205.60.10         to:1.1.2.2:23
SNAT       tcp  --  0.0.0.0/2            anywhere             to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            anywhere             to:1.1.2.2:21
SNAT       tcp  --  0.0.0.0/2            12.192.10.1          to:1.1.2.2
SNAT       udp  --  0.0.0.0/2            11.10.100.1          to:1.1.2.2:80
SNAT       tcp  --  125.125.100.0        anywhere             to:1.1.2.2
SNAT       tcp  --  30.20.10.40          anywhere             to:1.1.2.2:47
SNAT       tcp  --  205.200.30.11        120.10.20.30         to:1.1.2.2
SNAT       tcp  --  255.200.255.30       50.205.60.10         to:1.1.2.2:23
```

Par ce qu'on a considérer qu'aucun paquet n'est traité par le Firewall (d'après la solution proposée dans la Figure II.1.1), on remarque que les seules chaines remplis sont la chaine

FORWARD de la tables **Filter,** et les chaines **PREROUTING** et **POSTROUTING** de la table **Nat,** tandis que les chaines de la table **Mangle** ne sont pas touchés.

1.3. Conclusion:

Les résultats obtenus à partir des tests effectués montrent que ce préprocesseur fait son travail de la manière voulue, c'est-à-dire que l'appel des règles IPtables se fait correctement. Le noyau de Linux les ayant validé, comme cela est apparent dans la prise d'écran précédente, cela prouve qu'on est arrivé à la configuration cible du Firewall.

Conclusion générale:

Pour mieux répondre aux besoins de sécurité et pour éviter le piratage et le vol d'informations, surtout quand il s'agit d'informations sensibles, il faut sans cesse gérer et contrôler le flux d'informations qui transite d'un point à un autre.

Les Firewalls présentent effectivement un moyen de connexion et de filtrage sécurisé, mais ne répondent pas tout le temps à ce type de besoins, car la sécurité informatique est un domaine très vaste et varié. Pour cette raison, plusieurs moyens de sécurité sont actuellement combinés pour répondre aux nouvelles exigences de sécurité. Les Firewalls, qui sont des mécanismes de détection et de réaction aux intrusions, sont apparus pour répondre à ces besoins.

La configuration du Firewall Netfilter se fait par le bais de commandes IPtables muettes dont l'usage n'est pas commode. De fait, il s'imposait d'essayer de trouver un langage plus élégant pour configurer le Firewall. Ce dernier a été l'objet de notre travail. Nous avons ainsi conçu un langage permettant de configurer le Firewall et développé un préprocesseur permettant de traduire un source écrit dans notre langage en une suite de commandes IPtables équivalente.

Le travail accompli, bien que répondant au cahier de charges, pourrait être amélioré. Nous proposons la réalisation d'un éditeur intelligent qui utiliserait des jeux de couleurs pour identifier les différentes parties du source : mots réservés, identificateurs etc. et qui aurait pour charge de faire le contrôle syntaxique en parallèle et de proposer à l'utilisateur, au fur et à mesure de sa rédaction du programme source, la forme syntaxique à respecter, tel que cela est utilisé dans les langage de la Rapid Application Développement.

D'autres améliorations seraient envisageables. Il s'agit notamment de :

1- L'utilisation de XML pour modéliser les zones démilitarisées. Cette solution présenterait l'avantage d'utiliser XML qui est un standard en plus de la possibilité d'utiliser des parser XML pour la génération de code.

2- Généraliser la génération des règles de filtrages pour d'autres types de Firewall. Par exemple CISCO, Juniper, CheckPoint...

Concernant l'apport du projet à nos connaissances, nous pouvons dire qu'il nous a permis de comprendre le fonctionnement d'une manière précise du Firewall Netfilter ainsi que le fonctionnement des commandes IPtables et les DMZs avec leurs différentes architectures. Ce projet nous a permis également de mettre en pratique de nombreuses connaissances que nous avons acquises durant notre cursus d'ingénieur. Il nous a permis de nous familiariser avec un langage de programmation puissant et ouvert, à savoir le C, et aussi avec deux autres outils de génération de compilateurs que sont Lex et Yacc sous le système d'exploitation Linux.

Annexes

Annexe A
Utilisation de Lex et Yacc

1. Utilisation de Lex dans l'analyse lexicale [RII.1.1]

Lex est un générateur d'analyseurs lexicaux basés sur les expressions régulières. Le compilateur Lex vous permet de créer un programme qui pour chaque expression régulière trouvée pourra exécuter des commandes que vous aurez prédéfinies (écrites en C). Vous pouvez par exemple, très facilement, faire un programme qui reprend un texte et met en minuscules tous les mots qui sont en majuscule. Ou bien de mettre en majuscule toutes les premières lettres d'un mot qui est précédé d'un point, etc....

Le but de l'analyse lexicale est de transformer une suite de symboles en **terminaux** (un terminal peut être par exemple un nombre, un signe "+", un identificateur, etc...). Une fois cette transformation effectuée, la main est repassée à l'analyseur syntaxique.

Le but de l'analyseur lexical est donc de "consommer" des symboles et de les fournir à l'analyseur syntaxique.

Un fichier de description pour Lex est formé de trois parties, selon la structure suivante :

Déclarations
%%
Productions
%%
Code additionnel

Dans lequel aucune partie n'est obligatoire. Cependant, le premier **%%** l'est, afin d'indiquer la séparation entre les déclarations et les productions.

1.1. La première partie d'un fichier Lex

Cette partie d'un fichier Lex peut contenir :

- Du code écrit dans le langage cible, encadré par `%{` et `%}`, qui se retrouvera au début du fichier synthétisé par Lex. C'est ici que l'on va spécifier les fichiers à inclure. Lex recopie tel quel tout ce qui est écrit entre ces deux signes, qui devront toujours être placés en début de ligne.
- Des expressions régulières définissant des notions non terminales, telles que lettre, chiffre et nombre. Ces spécifications sont de la forme :

- `notion expression-régulière`. Les notions ainsi définies pourront alors être utilisées dans la suite de la première partie du fichier, ainsi que dans la deuxième partie, en les parenthésant par `{ et }`.

1.2. Écriture des expressions régulières

La construction de l'analyseur lexical « **lex** », nécessite une forme prédéfinie de la grammaire construite à partir de la table au dessus :

Table IV: La manière d'écriture des expressions régulières :

Symboles	Signification
x	le caractère "x"
.	n'importe quel caractère sauf \n
[xyz]	soit x, soit y, soit z
[^bz]	tous les caractères, **sauf** b et z
[a-z]	n'importe quel caractère entre a et z
[^a-z]	tous les caractères, **sauf** ceux compris entre a et z
R*	zéro R ou plus, ou R est n'importe quelle expression régulière
R+	un R ou plus
R?	zéro ou un R (c'est-à-dire un R optionnel)
R{2,5}	entre 2 et 5 R
R{2,}	2 R ou plus
R{2}	exactement 2 R
"[xyz\"foo"	la chaîne "[xyz"foo"
{NOTION}	l'expansion de la **NOTION** définie plus haut
\X	si X est un "a", "b", "f", "n", "r", "t", ou "v", représente l'interprétation ANSI-C de \X.
\0	le caractère ASCII 0
\123	le caractère dont le code ASCII est 123, **en octal**
\x2A	le caractère dont le code ASCII est 2A, **en hexadécimal**
RS	R suivi de S
R|S	R ou S
R/S	R, seulement s'il est suivi par S
^R	R, mais seulement en début de ligne

R$	R, mais seulement en fin de ligne
<<EOF>>	fin de fichier

Exemple :

Identificateur {lettre}(_|{lettre}|{chiffre10})*

Reconnaîtra comme identificateur les mots "integer", "une_variable", "a1", mais pas "ident" ni "1variable".

Enfin, comme dernier exemple, voici la définition d'un réel :

```
chiffre     [0-9]
entier      {chiffre}+
exposant    [eE] [+-]? {entier}
reel        {entier} ("."{entier})?{exposant}?
```

1.3. La deuxième partie d'un fichier Lex : les productions

Cette partie sert à indiquer à Lex ce qu'il devra faire lorsqu'il rencontrera telle ou telle notion. Celle-ci peut contenir :

- Des spécifications écrites dans le langage cible, encadrées par `%{` et `%}` (toujours placés en début de ligne), qui seront placées au début de la fonction **`yylex`**`()`, la fonction chargée de consommer les terminaux, et qui renvoi un entier.
- Des productions de la forme : `expression régulière action`. Si `action` est absente, Lex recopiera les caractères tels quels sur la sortie standard. Si action est présente, elle devra être écrite en langage cible. Si celle-ci comporte plus d'une seule instruction ou ne peut tenir sur une seule ligne, elle devra être parenthésée par `{` et `}`.

Il faut de plus savoir que les commentaires tels que /* ... */ ne peuvent être présents dans la deuxième partie d'un fichier Lex que s'ils sont placés dans les actions parenthésée. Dans le cas contraire, ceux-ci seraient considérés par Lex comme des expressions régulières ou des

actions, ce qui donnerait lieu à des messages d'erreur, ou au mieux à un comportement inattendu.

Enfin, la variable **yytext** désigne dans les actions les caractères acceptés par expression régulière. Il s'agit d'un tableau de caractères de longueur **yyleng** (donc défini comme char **yytext** [yyleng]).

1.4. Troisième partie d'un fichier Lex : le code additionnel

Tu peux mettre dans cette partie facultative tous les codes que tu veux. Si tu ne mets rien, Lex considère que c'est juste :

```
main () {
      yylex ();
       }
```

2. L'analyse grammaticale avec Yacc [RII.1.1]

'Lex' est un générateur d'analyseurs lexicaux, il ne peut pas effectuer une analyse grammaticale, c'est-à-dire que ' lex ' ne vous permet pas de saisir une grammaire complexe d'un langage et l'analyser, c'est le rôle de **'Yacc'**. 'Yacc' vous permet de faire ce que l'on appelle un 'parser'.

'Yacc' est un générateur d'analyseur grammatical, il a été spécialement conçu pour utiliser 'lex' comme analyseur lexical, ils marchent ensemble et chacun à son niveau. La partie 'lex' lit le fichier, décode les tokens et les renvoie à la partie 'Yacc'. La partie 'Yacc' ne lit pas le fichier en entrée mais seulement les tokens renvoyés par 'lex', au travers de la fonction 'yylex ()'. La communication entre les deux se passe aux niveaux des tokens et aussi de la variable 'yylval' que nous verrons plus tard.

Pour les puristes : Yacc (Yet Another Compiler Compiler) est un programme destiné compiler une grammaire du type ***LALR (1)*** et à produire le texte source d'un analyseur

syntaxique du langage engendré par cette grammaire. Il est aussi possible, en plus de la vérification de la syntaxe de la grammaire, de lui faire effectuer des actions sémantiques.

De la même manière que pour un fichier Lex, un fichier Yacc se compose de trois parties, de cette façon :

Déclarations

%%

Productions

%%

Code additionnel

Seul le premier séparateur **%%** et la deuxième partie étant obligatoires.

1.1. La première partie d'un fichier Yacc

La première partie d'un fichier Yacc peut contenir :

- Des spécifications écrites dans le langage cible, placées entre **%{** et **%}**, ces deux symboles étant obligatoirement en début de ligne.
- La déclaration des terminaux pouvant être rencontrés, grâce au mot-clé **%token**
- Le type de donnée du terminal courant, avec le mot-clé **%union**
- Des informations donnant la priorité et l'associativité des opérateurs.

L'axiome de la grammaire, avec le mot-clé **%start** (si celui-ci n'est pas précisé, l'axiome de la grammaire est la première production de la deuxième partie). La variable **yylval**, déclarée implicitement du type de **%union** a une importance fondamentale dans le fichier, puisque celle-ci contient la description du dernier terminal lu.

2.2. La deuxième section d'un fichier Yacc

Cette partie, qui ne peut pas être vide, contient :

- Des déclarations et/ou définitions encadrées par **%{** et **%}** ;
- Des productions de la grammaire du langage choisi.

Ces productions s'écrivent sous la forme générale :

```
notion_non_terminale:
corps_1        {action_sémantique_1}
| corps_2      {action_sémantique_2}
| ...
| corps_n      {action_sémantique_n}
;
```

Sachant que les corps_i peuvent être des notions terminales ou non terminales du langage.

2.3. La troisième partie d'un fichier Yacc

Cette partie, qui comporte le code additionnel, devra obligatoirement comporter une déclaration du **main ()** (qui devra appeler la fonction **yyparse ()**), et de la fonction **yyerror()**, appelée lorsqu'une erreur de syntaxe est trouvée.

Annexe B
Manuel d'utilisation

Manuel d'utilisation :

Ce logiciel est un outil d'aide à la conception et l'implémentation des Zones Démilitarisées (DMZ), il permet de configurer des Firewall a travers IPtables de Netfilter, et puisque IPtables est fiable et dispose de très nombreuses options qui permettent de faire du filtrage très fin, on peut atteint la configuration cible par la génération automatique de ces règles Iptables, et ça c'est l'objectif de ce logiciel.

La configuration du Firewall suit une syntaxe bien précise,et passe toujours par deux étapes :

1. Etape de déclaration : qui commence toujours par le mot clé 'DECLARATION', Elle contient la déclaration des attributs du Firewall tel que les interfaces et ses adresses IPs, une interface pour le réseau interne (LAN), une interface pour le réseau externe ou Internet (NET) et d'autres interfaces concernant les DMZ, cette déclaration à la structure suivante:

```
FW
{
...
}
```

Elle contient la déclaration des attributs du Firewall tel que les adresses réseaux, une adresse réseau pour le réseau local (LAN) et une ou plusieurs adresses réseaux pour les zones (DMZ), elle prend la structure suivante:

```
ZONS
{
...
}
```

Elle contient la déclaration des attributs des services tel que l'adresse IP, le port et protocole des zones (DMZ), elle prend la structure suivante:

```
SERVICES
{
...
}
```

Exemple explicatif:

```
DECLARATION
  FW
  {
  LAN : 220.102.200.2 , eth1
  ...
  ZONE 2 : 12.5.6.7 , eth3
  NET : 1.1.2.2 , eth4
  }
  ZONES
  {
  LAN : 1.4.5.10/8
  ZONE 1 : 4.8.6.55/2
  ...
  }
  SERVICES
  {
  WEB ZONE 1 : 1.2.3.5 , 80 , tcp
  WEB ZONE 2 : 4.4.4.5 , 5600 , udp
  ...
  }
```

2. Etape de configuration : qui commence toujours par le mot clé 'CONFIGURATION', elle contient des règles d'autorisation et de publication, autorisation des accès dans les différents sens et publication des services dans la DMZ. La syntaxe générale d'une telle règle doit prendre une forme de l'une de ces règles :

```
AUTORISER ACCEE : adresseIP LAN -> protocole NET
AUTORISER ACCEE : adresseIP LAN -> NumeroP , protocole NET
AUTORISER ACCEE : adresseIP  LAN -> adresseIP , protocole NET
AUTORISER ACCEE : LAN -> adresseIP  ,  protocole  NET
AUTORISER ACCEE : adresseIP LAN -> ZONE Numéro
AUTORISER ACCEE : adresseIP LAN -> adresseIP , NumeroP , protocole  NET
AUTORISER ACCEE : LAN -> NumeroP , protocole NET
AUTORISER ACCEE : LAN -> adresseIP , NumeroP , protocole NET
AUTORISER ACCEE : LAN -> adresseIP ZONE Numéro
 AUTORISER ACCEE : LAN -> protocole NET
 AUTORISER ACCEE : adresseIP LAN -> adresseIP ZONE Numéro
 AUTORISER ACCEE : LAN -> ZONE Numéro
```

PUBLIER IDFS ZONE Numéro

PUBLIER IDFS ZONE Numéro : NumeroP

L'utilisateur ne doit que remplacer :

- (adresseIP) par une adresse IP réel.
- (NumeroP) par de port correspond.
- (Numéro) par un numéro séconciel des zones.

Exemple explicatif:

CONFIGURATION

AUTORISER ACCEE : 20.2.10.1 LAN -> 200 , tcp NET

AUTORISER ACCEE : 10.0.5.6 LAN -> 100.21.80.12 , 302 , tcp NET

.

.

.

AUTORISER ACCEE : LAN -> ZONE 2

PUBLIER WEB ZONE 2

PUBLIER WEB ZONE 1 : 8080

Bibliographie

Sites Web:

[RI.1.1] : http://www.httr.ups-tlse.fr/exposes/TCP/tcp-1.3.html

[R I.1.2] : http://christian.caleca.free.fr/tcpip/adresse_ip.htm

[R I.2.1] : http://www.commentcamarche.net/protect/protintro.php3

[R I.2.2] : http://www.commentcamarche.net/protect/firewall.php3

[R I.2.3] : http://locoche.net/firewall.php

[R I.3.1] : http://fr.wikipedia.org/wiki/Netfilter

[R I.3.2] : www.ndlaprovidence.org/cours/btsig/debroise/formation%20reseau /proxyfirewall/Netfilter%20iptable%20.doc - Résultat complémentaire

[R I.4.1] : http://christian.caleca.free.fr/securite/les_firewalls.htm

[R I.4.2] : igm.univ-mlv.fr/~duris/NTREZO/20032004/Cheynet-DaSilva-Sebban-resentation-Firewalls.ppt

[RII.1.1] : http://pltplp.net/lex-yacc/lex.html.fr

Thèses :

[R 2.2] : Kehiliche.O;
« Conception et mise en œuvre d'un Firewall pour le contrôle d'accès à un réseau », Thèse de Magistère, Institut National de l'Informatique, 2004-2005

[CHA 2.1] : Chaouchi Hakima ;
« Conception et réalisation d'un outil intelligent d'aide ou choix de Firewall pour la protection d'un réseau informatique ».
Thèse de Magistère, Institut National de l'Informatique, 2000-2001